From formulas to cirquents in computability logic

Giorgi Japaridze[*]

Abstract

Computability logic (CoL) is a recently introduced semantical platform and ambitious program for redeveloping logic as a formal theory of computability, as opposed to the formal theory of truth which logic has more traditionally been. Its expressions represent interactive computational tasks seen as games played by a machine against the environment, and "truth" is understood as existence of an algorithmic winning strategy. With logical operators standing for operations on games, the formalism of CoL is open-ended, and has already undergone series of extensions. This article extends the expressive power of CoL in a qualitatively new way, generalizing *formulas* (to which the earlier languages of CoL were limited) to circuit-style structures termed *cirquents*. The latter, unlike formulas, are able to account for subgame/subtask sharing between different parts of the overall game/task. Among the many advantages offered by this ability is that it allows us to capture, refine and generalize the well known *independence-friendly logic* which, after the present leap forward, naturally becomes a conservative fragment of CoL, just as classical logic had been known to be a conservative fragment of the formula-based version of CoL. Technically, this paper is self-contained, and can be read without any prior familiarity with CoL.

MSC: primary: 03B47; secondary: 03F50; 03B70; 68Q10; 68T27; 68T30; 91A05

Keywords: Computability logic; Abstract resource semantics; Independence-friendly logic; Game semantics; Interactive computation

Contents

[*]This material is based upon work supported by the National Science Foundation under Grant No. 0208816

1 Introduction

Computability logic (CoL), introduced in [9, 14, 21], is a semantical platform and ambitious program for redeveloping logic as a formal theory of computability, as opposed to the formal theory of truth which logic has more traditionally been. Its expressions stand for interactive computational tasks seen as games played by a machine against the environment, and "truth" is understood as existence of an effective solution, i.e., of an algorithmic winning strategy.

With this semantics, CoL provides a systematic answer to the fundamental question "*what can be computed?*", just as classical logic is a systematic tool for telling what is true. Furthermore, as it turns out, in positive cases "*what* can be computed" always allows itself to be replaced by "*how* can be computed", which makes CoL of potential interest in not only theoretical computer science, but many more applied areas as well, including interactive knowledge base systems, resource oriented systems for planning and action, or declarative programming languages. On the logical side, CoL promises to be an appealing, constructive and computationally meaningful alternative to classical logic as a basis for applied theories. The first concrete steps towards realizing this potential have been made very recently in [23, 27], where CoL-based versions of Peano arithmetic were elaborated. The system constructed in [23] is an axiomatic theory of *effectively solvable* number-theoretic *problems* (just as the ordinary Peano arithmetic is an axiomatic theory of *true* number-theoretic *facts*); the system constructed in [26] is an axiomatic theory of *efficiently solvable* (namely, solvable in polynomial time) number-theoretic *problems*; in the same style, [27] constructs systems for polynomial space, elementary recursive, and primitive recursive computabilities. In all cases, a solution for a problem can be effectively — in fact, efficiently — extracted from a proof of it in the system, which reduces problem-solving to theorem-proving.

The formalism of CoL is open-ended. It has already undergone series of extensions ([20]-[24]) through introducing logical operators for new, actually or potentially interesting, operations on games, and this process will probably still continue in the future. The present work is also devoted to expanding the expressive power of CoL, but in a very different way. Namely, while the earlier languages of CoL were limited to *formulas*, this paper makes a leap forward by generalizing formulas to circuit-style structures termed *cirquents*. These structures, in a very limited form (compared with their present form), were introduced earlier [13, 19] in the context of the new proof-theoretic approach called *cirquent calculus*. Cirquent-based formalisms have significant advantages over formula-based ones, including exponentially higher efficiency and substantially greater expressive power. Both [13] and [19] pointed out the possibility and expediency of bringing cirquents into CoL. But a CoL-semantics for cirquents had never really been set up until now.

Unlike most of its predecessors, from the technical (albeit perhaps not philosophical or motivational) point of view, the present paper is written without assuming that the reader is already familiar with the basic concepts and techniques of computability logic. It is organized as follows.

Section 2 reintroduces the concept of games and interactive computability on which the semantics of CoL is based. A reader familiar with the basics of CoL may safely skip this section.

Section 3 introduces the simplest kind of cirquents, containing only the traditional $\vee$ and $\wedge$ sorts of gates (negation, applied directly to inputs, is also present). These are nothing but (possibly infinite) Boolean circuits in the usual sense, and the semantics of CoL for them coincides with the traditional, classical semantics. While such cirquents — when finite — do not offer higher expressiveness than formulas do, they *do* offer dramatically higher efficiency. This fact alone, in our days of logic being increasingly CS-oriented, provides sufficient motivation for considering a switch from formulas to cirquents in logic, even if we are only concerned with classical logic. The first steps in this direction have already been made in [19], where a cirquent-based sound and complete deductive system was set up for classical logic. That system was shown to provide an exponential speedup of proofs over its formula-based counterparts.

Each of the subsequent Sections 4-8 conservatively generalizes the cirquents and the semantics of the preceding sections.

Section 4 strengthens the expressiveness of cirquents by allowing new, so called *selectional*, sorts of gates, with the latter coming in three — *choice* $\sqcup, \sqcap$, *sequential* $\triangledown, \vartriangle$ and *toggling* $\curlyvee, \curlywedge$ — flavors of disjunction and conjunction. Unlike $\vee$ and $\wedge$ which stand for parallel combinations of (sub)tasks, selectional gates model decision steps in the process of interaction of the machine with its environment. Cirquents with such gates, allowing us to account for the possibility of *sharing* nontrivial subgames/subtasks, have never been considered in the past. They, even when finite, are more expressive (let alone efficient) than formulas with selectional connectives.

Section 5 introduces the idea of *clustering* selectional gates. Clusters are, in fact, generalized gates —

switch-style devices that permit to select one of several n-tuples of inputs and connect them, in a parallel fashion, with the n-tuple of outputs, with ordinary gates being nothing but the special cases of clusters where $n = 1$. Clustering makes it possible to express a new kind of sharing, which can be characterized as "sharing children without sharing grandchildren" — that is, sharing decisions (associated with child gates) without sharing the results of such decisions (the children of those gates). The ability to account for this sort of sharing yields a further qualitative increase in the expressiveness of the associated formalism.

Sections 6 and 7 extend clustering from selectional gates to the traditional sorts $\vee, \wedge$ of gates. It turns out that the resulting cirquents — even without selectional gates — are expressive enough to fully capture the well known and extensively studied *independence-friendly (IF) logic* introduced by Hintikka and Sandu [5]. At the same time, cirquents with clustered $\vee, \wedge$-gates yield substantially higher expressiveness than IF logic does. Due to this fact, they overcome a series of problems emerging within the earlier known approaches to IF logic. One of such problems is the inability of the traditional formalisms of IF logic to account for independence from conjunctions and disjunctions in the same systematic way as they account for independence from quantifiers. Correspondingly, attempts to develop IF logic at the purely propositional level have not been able to go beyond certain very limited and special syntactic fragments of the language. In contrast, our approach saves classical logic's nice phenomenon of quantifiers being nothing but "long" conjunctions and disjunctions, so that one can do with the latter everything that can be done with the former, and vice versa. As a result, we can now (at last) meaningfully talk about *propositional IF logic* in the proper sense without any unsettling syntactic restrictions. Another problem arising with IF logic is the "unplayability" (cf. [31]) of the incomplete-information games traditionally associated with its formulas, as such games violate certain natural game-theoretic principles such as *perfect recall*. Attempts to associate reasonable game-theoretic intuitions with imperfect-information games typically have to resort to viewing the two parties not as individual players but rather as teams of cooperating but non-communicating players. This approach, however, may often get messy, and tends to give rise to series of new sorts of problems. Väänänen [34] managed to construct a semantics for IF logic based on perfect-information games. This, however, made games unplayable for a new reason: moves in Väänänen's games are second-order objects and hence are "unmakable". Our approach avoids the need to deal with imperfect-information, second-order-move, or multiple-player games altogether.

Section 7 also introduces a further generalization of cirquents through what is termed *ranking*. Cirquents with ranking (and with only $\vee, \wedge$ gates) allow us to further capture the so called *extended IF logic* (cf. [32]) but, again, are substantially more expressive than the latter. They also overcome one notable problem arising in extended IF logic, which is the (weak) negation's not being able to act as an ordinary connective that can be meaningfully applied anywhere within a formula.

Section 8 extends the formalism of cirquents by allowing additional sorts of inputs termed *general*, as opposed to the *elementary* inputs to which the cirquents of the earlier sections are limited. Unlike elementary inputs that are interpreted just as $\top$ ("true") or $\perp$ ("false"), general inputs stand for any, possibly nontrivial, games. This way, cirquents become and can be viewed as a (complex) *operations* on games, only very few of which are expressible through formulas.

Section 9 sets up an alternative semantics for cirquents termed *abstract resource semantics* (ARS). It is a companion and faithful technical "assistant" of CoL. ARS also has good claims to be a materialization and generalization of the resource intuitions traditionally — but somewhat wrongly — associated with linear logic and its variations. Unlike CoL, ARS has already met cirquents in the past, namely, in [13, 19]. The latter, however, unlike the present paper, elaborated ARS only for a very limited class of cirquents — cirquents with just $\vee, \wedge$ gates and without clustering or ranking.

Section 10 proves that CoL and ARS validate exactly the same cirquents. Among the expected applications of this theorem is facilitating soundness/completeness proofs for various deductive systems for (fragments of) CoL, as technically it is by an order of magnitude easier to deal with (the simple and "naive") ARS than to deal with (the complex and "serious") CoL directly.

2 Games

As noted, computability logic understands interactive computational problems as games played between two players: *machine* and *environment*. The symbolic names for these two players are $\top$ and $\perp$, respectively. $\top$ is a deterministic mechanical device (thus) only capable of following algorithmic strategies, whereas there are no restrictions on the behavior of $\perp$, which represents a capricious user, the blind forces of nature, or the devil himself. Our sympathies are with $\top$, and by just saying "won" or "lost" without specifying a player, we always mean won or lost by $\top$. $\wp$ is always a variable ranging over $\{\top, \bot\}$, with $\neg\wp$ meaning $\wp$'s adversary, i.e. the player which is not $\wp$.

Before getting to a formal definition of games, we agree that a **move** is always a finite string over the standard keyboard alphabet. A **labeled move** (**labmove**) is a move prefixed with $\top$ or $\bot$, with its prefix (**label**) indicating which player has made the move. A **run** is a (finite or infinite) sequence of labmoves, and a **position** is a finite run. Runs will be often delimited by "$\langle$" and "$\rangle$", with $\langle\rangle$ thus denoting the **empty run**.

The following is a formal definition of the concept of a game, combined with some less formal conventions regarding the usage of certain terminology. It should be noted that the concept of a game considered in CoL is more general than the one defined below, with games in our present sense called *constant games*. Since we (for simplicity) only consider constant games in this paper, we omit the word "constant" and just say "game".

Definition 2.1 A **game** is a pair $A = (\mathbf{Lr}^A, \mathbf{Wn}^A)$, where:

1. $\mathbf{Lr}^A$ is a set of runs satisfying the condition that a finite or infinite run is in $\mathbf{Lr}^A$ iff all of its nonempty finite — not necessarily proper — initial segments are in $\mathbf{Lr}^A$ (notice that this implies $\langle\rangle \in \mathbf{Lr}^A$). The elements of $\mathbf{Lr}^A$ are said to be **legal runs** of A, and all other runs are said to be **illegal runs** of A. We say that α is a **legal move** for $\wp$ in a position Φ of A iff $\langle\Phi, \wp\alpha\rangle \in \mathbf{Lr}^A$; otherwise α is an **illegal move**. When the last move of the shortest illegal initial segment of Γ is $\wp$-labeled, we say that Γ is a **$\wp$-illegal run** of A.

2. $\mathbf{Wn}^A$ is a function that sends every run Γ to one of the players $\top$ or $\bot$, satisfying the condition that if Γ is a $\wp$-illegal run of A, then $\mathbf{Wn}^A\langle\Gamma\rangle = \neg\wp$.[1] When $\mathbf{Wn}^A\langle\Gamma\rangle = \wp$, we say that Γ is a **$\wp$-won** (or **won by** $\wp$) run of A; otherwise Γ is **lost by** $\wp$. Thus, an illegal run is always lost by the player who has made the first illegal move in it.

It is clear from the above definition that, when defining a particular game A, it would be sufficient to specify what *positions* (finite runs) are legal, and what *legal runs* are won by $\top$. Such a definition will then uniquely extend to all — including infinite and illegal — runs. We will implicitly rely on this observation in the sequel.

A game is said to be **elementary** iff it has no legal runs other than (the always legal) empty run $\langle\rangle$. That is, an elementary game is a "game" without any (legal) moves, automatically won or lost. There are exactly two of such games, for which we use the same symbols $\top$ and $\bot$ as for the two players: the game $\top$ automatically won by player $\top$, and the game $\bot$ automatically won by player $\bot$.[2] Computability logic is a conservative extension of classical logic, understanding classical propositions as elementary games. And, just as classical logic, it sees no difference between any two true propositions such as "$0 = 0$" and "*Snow is white*", and identifies them with the elementary game $\top$; similarly, it treats false propositions such as "$0 = 1$" or "*Snow is black*" as the elementary game $\bot$.

The *negation* $\neg A$ of a game A is understood as the game obtained from A by interchanging the roles of the two players, i.e., making $\top$'s (legal) moves and wins $\bot$'s moves and wins, and vice versa. Precisely, let us agree that, for a run Γ, $\neg\Gamma$ means the result of changing in Γ each label $\top$ to $\bot$ and vice versa. Then:

Definition 2.2 The **negation** $\neg A$ of a game A is defined by stipulating that, for any run Γ,

- $\Gamma \in \mathbf{Lr}^{\neg A}$ iff $\neg\Gamma \in \mathbf{Lr}^A$;
- $\mathbf{Wn}^{\neg A}\langle\Gamma\rangle = \top$ iff $\mathbf{Wn}^A\langle\neg\Gamma\rangle = \bot$.

Obviously the negation of an elementary game is also elementary. Generally, when applied to elementary games, the meaning of $\neg$ fully coincides with its classical meaning. So, $\neg$ is a conservative generalization of classical negation from elementary games to all games.

Note the relaxed nature of our games. They do not impose any regulations on when either player can or should move. This is entirely up to the players. Even if we assume that illegal moves physically cannot be made, it is still possible that in certain (or all) positions both of the players have legal moves, and then the next move will be made (if made at all) by the player who wants or can act sooner. This brings us to the next question to clarify: how are our games really played, and what does a *strategy* mean here?

[1] We write $\mathbf{Wn}^A\langle\Gamma\rangle$ for $\mathbf{Wn}^A(\Gamma)$.

[2] Precisely, we have $\mathbf{Wn}^{\top}\langle\rangle = \top$ and $\mathbf{Wn}^{\bot}\langle\rangle = \bot$.

In traditional game-semantical approaches, including those of Lorenzen [28], Hintikka [4] or Blass [2], player's strategies are understood as *functions* — typically as functions from interaction histories (positions) to moves, or sometimes (Abramsky and Jagadeesan [1]) as functions that only look at the latest move of the history. This *strategies-as-functions* approach, however, is inapplicable in the context of computability logic, whose relaxed semantics, in striving to get rid of any "bureaucratic pollutants" and only deal with the remaining true essence of games, has no structural rules and thus does not regulate the order of moves. As noted, here often either player may have (legal) moves, and then it is unclear whether the next move should be the one prescribed by $\top$'s strategy function or the one prescribed by the strategy function of $\bot$. In fact, for a game semantics whose ambition is to provide a comprehensive, natural and direct tool for modeling interactive computations, the strategies-as-functions approach would be less than adequate, even if technically possible. This is so for the simple reason that the strategies that real computers follow are not functions. If the strategy of your personal computer was a function from the history of interaction with you, then its performance would keep noticeably worsening due to the need to read the continuously lengthening — and, in fact, practically infinite — interaction history every time before responding. Fully ignoring that history and looking only at your latest keystroke in the spirit of [1] is also certainly not what your computer does, either. The inadequacy of the strategies-as-functions approach becomes especially evident when one tries to bring computational complexity issues into interactive computation, the next natural target towards which CoL has already started making its first steps ([25, 26, 27]).

In computability logic, ($\top$'s effective) strategies are defined in terms of interactive machines, where computation is one continuous process interspersed with — and influenced by — multiple "input" (environment's moves) and "output" (machine's moves) events. Of several, seemingly rather different yet equivalent, machine models of interactive computation studied in CoL, here we will consider the most basic, **HPM** ("Hard-Play Machine") model.

An HPM is nothing but a Turing machine with the additional capability of making moves. The adversary can also move at any time, with such moves being the only nondeterministic events from the machine's perspective. Along with the ordinary read/write *work tape*, the machine also has an additional tape[3] called the *run tape*. The latter, at any time, spells the "current position" of the play. The role of this tape is to make the interaction history fully visible to the machine. It is read-only, and its content is automatically updated every time either player makes a move.

In these terms, an **algorithmic solution** ($\top$'s **winning strategy**) for a given game A is understood as an HPM $\mathcal{M}$ such that, no matter how the environment acts during its interaction with $\mathcal{M}$ (what moves it makes and when), the run incrementally spelled on the run tape is a $\top$-won run of A. When this is the case, we say that $\mathcal{M}$ **wins**, or **solves**, A, and that A is a **computable**, or **algorithmically solvable**, game.

As for $\bot$'s strategies, there is no need to define them: all possible behaviors by $\bot$ are accounted for by the different possible nondeterministic updates of the run tape of an HPM.

In the above outline, we described HPMs in a relaxed fashion, without being specific about technical details such as, say, how, exactly, moves are made by the machine, how many moves either player can make at once, what happens if both players attempt to move "simultaneously", etc. As it turns out, all reasonable design choices yield the same class of winnable games as long as we consider a certain natural subclass of games called *static*.

Intuitively, static games are interactive tasks where the relative speeds of the players are irrelevant, as it never hurts a player to postpone making moves. In other words, they are games that are contests of intellect rather than contests of speed. And one of the theses that computability logic philosophically relies on is that static games present an adequate formal counterpart of our intuitive concept of "pure", speed-independent interactive computational problems. Correspondingly, computability logic restricts its attention (more specifically, possible interpretations of the atoms of its formal language) to static games. Below comes a formal definition of this concept.

For either player $\wp$, we say that a run Υ is a **$\wp$-delay** of a run Γ iff:

- for both players $\wp' \in \{\top, \bot\}$, the subsequence of $\wp'$-labeled moves of Υ is the same as that of Γ, and
- for any $n, k \geq 1$, if the nth $\wp$-labeled move is made later than (is to the right of) the kth $\neg\wp$-labeled move in Γ, then so is it in Υ.

[3]An HPM often also has a third tape called the *valuation tape*. Its function is to provide values for the variables on which a game may depend. However, as we remember, in this paper we only consider constant games — games that do not depend on any variables. This makes it possible to safely remove the valuation tape from the HPM model (or leave it there but fully ignore), as this tape is no longer relevant.

The above conditions mean that in Υ each player has made the same sequence of moves as in Γ, only, in Υ, $\wp$ might have been acting with some delay.

Let us say that a run is **$\wp$-legal** iff it is not $\wp$-illegal. That is, a $\wp$-legal run is either simply legal, or the player responsible for (first) making it illegal is $\neg\wp$ rather than $\wp$.

Now, we say that a game A is **static** iff, whenever a run Υ is a $\wp$-delay of a run Γ, we have:

- if Γ is a $\wp$-legal run of A, then so is Υ;[4]
- if Γ is a $\wp$-won run of A, then so is Υ.

The class of static games is closed under all game operations studied in CoL, and all games that we shall see in this paper are static. Throughout this paper, we use the term "**computational problem**", or simply "**problem**", is a synonym of "static game".

A simplest example of a non-static game would be the game where all moves are legal, and which is won by the player who moves first. The chances of a player to succeed only depend on its relative speed, that is. Such a game hardly represents any meaningful computational problem.

3 The simplest kind of cirquents

We fix some infinite collection of finite alphanumeric expressions called **atoms**, and use p, q, r, s, p_1, p_2, $p(3,6)$, $q_7(1,1,8)$, ... as metavariables for them. If p is an atom, then the expressions p and $\neg p$ are said to be **literals**.

Let us agree that, in this section, by a **graph** we mean a directed acyclic multigraph with countably many nodes and edges, where the outgoing edges of each node are arranged in a fixed left-to-right order (edges #1, #2, etc.), and where each node is labeled with either a literal or $\vee$ or $\wedge$. Since the sets of nodes and edges are countable, we assume that they are always subsets of $\{1, 2, 3, \ldots\}$. For a node n of the graph, the string representing n in the standard decimal notation is said to be the **ID number**, or just the **ID**, of the node. Similarly for edges.

The nodes labeled with $\wedge$ or $\vee$ we call **gates**, and the nodes labeled with literals we call **ports**. Specifically, a node labeled with a literal L is said to be a an **L-port**; a $\wedge$-labeled node is said to be a **$\wedge$-gate**; and a $\vee$-labeled node is said to be a **$\vee$-gate**. When there is an edge from a node a to a node b, we say that b is a **child** of a and a is a **parent** of b. The relations "**descendant**" and "**ancestor**" are the transitive closures of the relations "child" and "parent", respectively. The meanings of some other standard relations such as "grandchild", "grandparent", etc. should also be clear.

The **outdegree** of a node of a graph is the quantity of outgoing edges of that node, which can be finite or infinite. Since there are only countably many edges, any two infinite outdegrees are equal. Similarly, the **indegree** of a node is the quantity of the incoming edges of that node.

We say that a graph is **well-founded** iff there is no infinite sequence $a_1, a_2, a_3, \ldots$ of nodes where each a_i is a predecessor of a_{i+1}. Of course, any (directed acyclic) graph with finitely many nodes is well-founded.

We say that a graph is **effective** iff so are the following basic predicates and partial functions characterizing it: "*x is a node*", "*x is an edge*", "*the label of node x*", "*the outdegree of node x*", "*the y'th outgoing edge of node x*", "*the origin of edge x*", "*the destination of edge x*".

Definition 3.1 A **cirquent** is an effective, well-founded graph satisfying the following two conditions:

1. Ports have no children.
2. There is a node, called the **root**, which is an ancestor of all other nodes in the graph.

We say that a cirquent is **finite** iff it has only finitely many edges (and hence nodes); otherwise it is **infinite**.

We say that a cirquent is **tree-like** iff the indegree of each of its non-root node is 1.

Graphically, we represent ports through the corresponding literals, $\vee$-gates through $\vee$-inscribed circles, and $\wedge$-gates through $\wedge$-inscribed circles. We agree that the direction of an edge is always upward, which

[4]In most papers on CoL, the concept of static games is defined without this (first) condition. In such cases, however, the existence of an always-illegal move $\spadesuit$ is stipulated in the definition of games. The first condition of our present definition of static games turns out to be simply derivable from that stipulation. This and a couple of other minor technical differences between our present formulations from those given in other pieces of literature on CoL only signify presentational and by no means conceptual variations.

allows us to draw lines rather than arrows for edges. It is understood that the official order of the outgoing edges of a gate coincides with the (left to right) order in which the edges are drawn. Also, typically we do not indicate the IDs of nodes unless necessary. In most cases, what particular IDs are assigned to the nodes of a cirquent is irrelevant and such an assignment can be chosen arbitrarily. Similarly, edge IDs are usually irrelevant, and they will never be indicated.

Below are a few examples of cirquents. Note that cirquents may contain parallel edges, as we agreed that "graph" means "multigraph". Note also that not only ports but also gates can be childless.

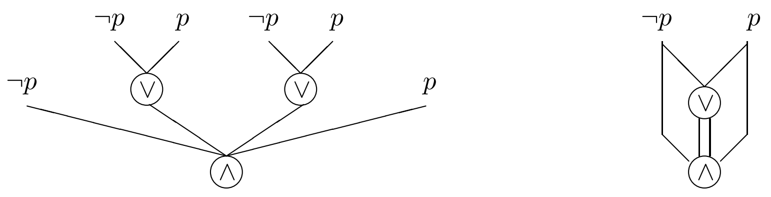


Figure 1: Finite cirquents

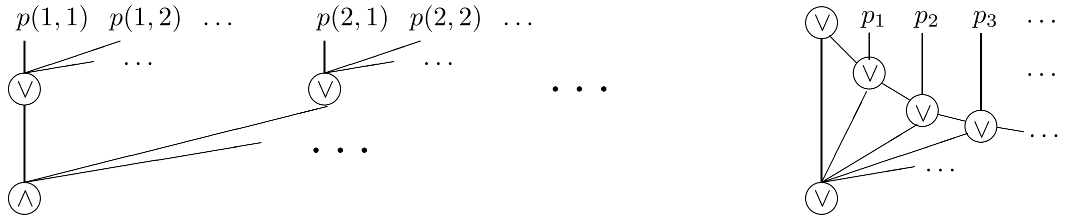


Figure 2: Infinite cirquents

By an **interpretation**[5] in this and the following few sections we mean a function * that sends each atom p to one of the values (elementary games) $p^* \in \{\top, \bot\}$. It immediately extends to a mapping from all literals to $\{\top, \bot\}$ by stipulating that $(\neg p)^* = \neg(p^*)$; that is, * sends $\neg p$ to $\top$ iff it sends p to $\bot$.

Each interpretation * induces the predicate of *truth* with respect to (w.r.t.) * for cirquents and their nodes, as defined below. This definition, as well as similar definitions given later, silently but essentially relies on the fact that the graphs that we consider are well-founded.

Definition 3.2 Let C be a cirquent and * an interpretation. With "port" and "gate" below meaning those of C, and "true" to be read as "**true w.r.t.** *", we say that:

- An L-port is true iff $L^* = \top$ (any literal L).
- A $\vee$-gate is true iff so is at least one of its children (thus, a childless $\vee$-gate is never true).
- A $\wedge$-gate is true iff so are all of its children (thus, a childless $\wedge$-gate is always true).

Finally, we say that the cirquent C is true iff so is its root.

Definition 3.3 Let C be a cirquent and * an interpretation. We define C^* to be the elementary game the (only) legal run $\langle\rangle$ of which is won by $\top$ iff C is true w.r.t. *.

Thus, every interpretation * "interprets" a cirquent C as one of the elementary games $\top$ or $\bot$. This will not necessarily be the case for the more general sorts of cirquents introduced in later sections though.

We say that two cirquents C_1 and C_2 are **extensionally identical** iff, for every interpretation *, $C_1^* = C_2^*$. For instance, the two cirquents of Figure 1 are extensionally identical. Occasionally we may say "*equivalent*" instead of "extensionally identical", even though one should remember that equivalence often (in other pieces of literature) may mean something weaker than extensional identity.

[5]This concepts is termed "perfect interpretation" in the other pieces of literature on CoL, where the word "interpretation" is reserved for a slightly more general concept. Since we only deal with perfect interpretations in this paper, we omit the word "perfect" and just say "interpretation".

It should be pointed out that the above definition of extensional identity applies not only to cirquents in the sense of the present section. It extends, without any changes in phrasing, to cirquents in the more general sense of any of the subsequent sections of this paper as well.

Finally, we say that a cirquent C is **valid** iff, for any interpretation *, $C^* = \top$.

Obviously the semantics that we have defined in this section is nothing but the kind old semantics of classical logic. Computability logic fully agrees with and adopts the latter. This is exactly what makes computability logic a conservative extension of classical logic.

Let us agree that, whenever we speak about formulas of classical logic, they are assumed to be written in *negation normal form*, that is, in the form where the negation symbol $\neg$ is only applied to atoms. If an expression violates this condition, it is to be understood just as a standard abbreviation. Similarly, if we write $E \rightarrow F$, it is to be understood as an abbreviation of $\neg E \vee E$. Also, slightly deviating from the tradition, we allow any finite numbers of arguments for conjunctions and disjunctions in classical formulas. The symbol $\top$ will be understood as an abbreviation of the empty conjunction, $\perp$ as an abbreviation of the empty disjunction, the expression $\wedge\{E\}$ will be used for the conjunction whose single conjunct is E, and similarly for $\vee\{E\}$. More generally, for any $n \geq 0$, $\wedge\{E_1, \ldots, E_n\}$ can (but not necessarily will) be written instead of $E_1 \wedge \ldots \wedge E_n$, and similarly for $\vee\{E_1, \ldots, E_n\}$.

Every formula of classical propositional logic then can and will be seen as a finite tree-like cirquent, namely, the cirquent which is nothing but the parse tree for that formula. For instance, we shall understand the formula $\neg p \wedge (\neg p \vee p) \wedge (\neg p \vee p) \wedge p$ as the left cirquent of Figure 1.

Every finite — not necessarily tree-like — cirquent can also be translated into an equivalent formula of classical propositional logic. This can be done by first turning the cirquent into an extensionally identical tree-like cirquent by duplicating and separating shared nodes, and then writing the formula whose parse tree such a cirquent is. For instance, applying this procedure to the right cirquent of Figure 1 turns it into the left cirquent of the same figure and thus the formula $\neg p \wedge (\neg p \vee p) \wedge (\neg p \vee p) \wedge p$.

Without loss of generality, we assume that, unless otherwise specified, the universe of discourse in all cases that we consider — i.e., the set over which the variables of classical first order logic range — is $\{1, 2, 3, \ldots\}$. Then, from the perspective of classical first order logic, the universal quantification of $E(x)$ is nothing but the "long conjunction" $E(1) \wedge E(2) \wedge E(3) \wedge \ldots$, and the existential quantification of $E(x)$ is nothing but the "long disjunction" $E(1) \vee E(2) \vee E(3) \vee \ldots$. To emphasize this connection, let us agree to use the expression $\bigwedge x E(x)$ for the former and $\bigvee x E(x)$ for the latter, instead of the more usual $\forall x E(x)$ and $\exists x E(x)$ (computability logic reserves $\forall x$ and $\exists x$ for another, so called *blind*, sort of quantifiers; semantically they, just like $\bigwedge x$ and $\bigvee x$, are conservative generalizations of the classical quantifiers).

Since quantifiers are conjunctions or disjunctions, it is obvious that all formulas of classical first order logic can also be seen as tree-like (albeit no longer finite) cirquents. For instance, the formula $\bigwedge x \bigvee y\ p(x, y)$ is nothing but the left cirquent of Figure 2.

On the other hand, unlike the case with finite cirquents, obviously not all infinite cirquents can be directly and adequately translated into formulas of classical logic. Of course, the great expressive power achieved by infinite cirquents, by itself, does not mean much, because such cirquents are generally "unwritable" or "undrawable". The cirquents of Figure 2 are among the not many lucky exceptions, but even there, the usage of the ellipsis is very informal, relying on a human reader's ability to see patterns and generalize. In order to take advantage of the expressiveness of cirquents, one needs to introduce notational conventions allowing to represent certain infinite patterns and (sub)cirquents through finite means. The quantifiers $\bigwedge x$ and $\bigvee x$ are among such means. Defining new, ever more expressive means (not reducible to $\bigwedge x$, $\bigvee x$) is certainly possible. Among the advantages of considering all — rather than only finitely represented — cirquents as we do in this paper is that a semantics for them has to be set up only once. If and when various abbreviations and finite means of expression are introduced, one will only need to explain what kinds of cirquents or subcirquents they stand for, without otherwise redefining or extending the already agreed-upon general semantics for cirquents.

But higher expressiveness is not the only advantage of cirquents over formulas. Another very serious advantage is the higher efficiency of cirquents. Let us for now talk only about finite cirquents. As we know, all such cirquents can be written as formulas of classical propositional logic. So, they do not offer any additional expressive power. But they *do* offer dramatically improved efficiency in representing Boolean functions. In Section 8 of [19] one can find examples of naturally emerging sets of Boolean functions which can be represented through polynomial size cirquents but require exponential sizes if represented through formulas. The higher efficiency of cirquents is achieved due to the possibility to *share* children between different parents — the mechanism absent in formulas, because of which an exponential explosion may easily occur when translating a (non-tree-like) cirquent into an equivalent formula. Imagine where the computer industry would be at present if, for some strange reason, the computer engineers had insisted on tree-like (rather than graph-like) circuitries!

Cirquents offer not only improved efficiency of expression, but also improved efficiency of proofs in deductive systems. In [19], a cirquent-based analytic deductive system **CL8** for classical propositional logic is set up which is shown to yield an exponential improvement (over formula-based analytic deductive systems) in proof efficiency. For instance, the notorious propositional Pigeonhole principle, which is known to have no polynomial size analytic proofs in traditional systems, has been shown to have such proofs in **CL8**.

4 Selectional gates

We now start a series of generalizations for cirquents and their semantics. We agree that, throughout the rest of this paper, unless otherwise specified, "cirquent" and all related terms are always to be understood in their latest, most general, sense.

In this section we extend the concept of **cirquents** defined in the previous section by allowing, along with $\vee$ and $\wedge$, the following six additional possible labels for gates: $⩛, ⩚, \bigtriangledown, \bigtriangleup, \sqcup, \sqcap$. Gates labeled with any of these new symbols we call **selectional** gates. And gates labeled with the old $\vee$ or $\wedge$ we call **parallel** gates.

Selectional gates, in turn, are subdivided into three groups:

- $\{⩛, ⩚\}$, referred to as **toggling** gates;
- $\{\bigtriangledown, \bigtriangleup\}$, referred to as **sequential** gates;
- $\{\sqcup, \sqcap\}$, referred to as **choice** gates.

The eight kinds of gates are also divided into the following two groups:

- $\{\vee, ⩛, \bigtriangledown, \sqcup\}$, termed **disjunctive** gates (or simply **disjunctions**);
- $\{\wedge, ⩚, \bigtriangleup, \sqcap\}$, termed **conjunctive** gates (or simply **conjunctions**).

Thus, $\wedge$ is to be referred to as "parallel conjunction", $\sqcup$ as "choice disjunction", etc.

The same eight symbols can be used to construct formulas in the standard way. Of course, in the context of formulas, these symbols will be referred to as **operators** or **connectives** rather than as gates. Formulas of computability logic accordingly may also use eight sorts of **quantifiers**. They are written as symbols of the same shape as the corresponding connectives, but in a larger size. Two of such quantifiers — $\bigwedge x$ and $\bigvee x$ — have already been explained in the previous section. The remaining quantifiers are understood in the same way: $\bigsqcap xE(x)$ is the infinite choice conjunction $E(1) \sqcap E(2) \sqcap \ldots$, $\bigsqcup xE(x)$ is the infinite choice disjunction $E(1) \sqcup E(2) \sqcup \ldots$, and similarly for $⩚, ⩛, \bigtriangleup, \bigtriangledown$. Since quantifiers are again nothing but "long" conjunctions and disjunctions, as pointed out in the previous section, there is no necessity to have special gates for them, as our approach that permits gates with infinite outdegrees covers them all. For similar reasons, there is no necessity to have special gates for what computability logic calls *(co)recurrence operators*. The *parallel recurrence* $⅄E$ of E is defined as the infinite parallel conjunction $E \wedge E \wedge \ldots$, the *parallel corecurrence* $\curlyvee E$ of E is defined as the infinite parallel disjunction $E \vee E \vee \ldots$, and similarly for *toggling recurrence* $⅄$, *toggling corecurrence* $\curlyvee$, *sequential recurrence* $⅄$ and *sequential corecurrence* $\curlyvee$ (choice recurrence and corecurrence are not considered because, semantically, both $E \sqcap E \sqcap \ldots$ and $E \sqcup E \sqcup \ldots$ are simply equivalent to E).

All of the operators $\vee, \wedge, ⩛, ⩚, \bigtriangledown, \bigtriangleup, \sqcup, \sqcap$, including their quantifier and recurrence versions, have been already motivated, defined and studied in computability logic. This, however, has been done only in the context of formulas. In this paper we extend the earlier approach and concepts of computability logic from formulas to cirquents as generalized formulas.

As before, an interpretation is an assignment * of $\top$ or $\bot$ to each atom, extended to all literals by commuting with $\neg$. And, as before, such an assignment induces a mapping that sends each cirquent C to a game C^*. When C is a cirquent in the sense of the previous section, C^* is always an elementary game ($\top$ or $\bot$). When, however, C contains selectional gates, the game C^* is no longer elementary.

To define such games C^*, let us agree that, throughout this papers, positive integers are identified with their decimal representations, so that, when we say "number n", we may mean either the *number* n as an abstract object, or the *string* that represents n in the decimal notation. Among the benefits of this convention is that it allows us to identify nodes of a cirquent with their IDs.

Definition 4.1 Let C be a cirquent, * an interpretation, and Φ a position. Φ is a **legal position** of the game C^* iff, with a "gate" below meaning a gate of C, the following conditions are satisfied:

1. Each labmove of Φ has one of the following forms:
   (a) $\top g.i$, where g is a $\curlyvee$-, $\bigtriangledown$- or $\sqcup$-gate and i is a positive integer not exceeding the outdegree of g;
   (b) $\bot g.i$, where g is a $\curlywedge$-, $\bigtriangleup$- or $\sqcap$-gate and i is a positive integer not exceeding the outdegree of g.
2. Whenever g is a choice gate, Φ contains at most one occurrence of a labmove of the form $\wp g.i$.
3. Whenever g is a sequential gate and Φ is of the form $\langle \ldots, \wp g.i, \ldots, \wp g.j, \ldots \rangle$, we have $i < j$.

Note that the set of legal runs of C^* does not depend on *. Hence, in the sequel, we can unambiguously omit "*" and simply say "legal run of C".

The intuitive meaning of a move of the form $g.i$ is **selecting** the ith outgoing edge — together with the child pointed at by such an edge — of (in) the selectional gate g. In a disjunctive selectional gate, a selection is always made by player $\top$; and in a conjunctive selectional gate, a selection is always made by player $\bot$. The difference between the three types of selectional gates is only in how many selections and in what order are allowed to be made in the same gate. In a choice gate, a selection can be made only once. In a sequential gate, selections can be reconsidered any number of times, but only in the left-to-right order (once an edge $\#i$ is selected, no edge $\#j$ with $j \leq i$ can be (re)selected afterwards). In a toggling gate, selections can be reconsidered any number of times and in any order. This includes the possibility to select the same edge over and over again.

Definition 4.2 In the context of a given cirquent C and a legal run Γ of C, we will say that a selectional gate g is **unresolved** iff either no moves of the form $g.j$ have been made in Γ, or infinitely many such moves have been made.[6] Otherwise g is **resolved** and, where $g.i$ is the last move of the form $g.j$ made in Γ, the child pointed at by the ith outgoing edge of g is said to be the **resolvent** of g.

Intuitively, the resolvent is the "final", or "ultimate" selection of a child made in gate g by the corresponding player. There is no such "ultimate" selection in unresolved gates.

The following definition conservatively generalizes Definition 3.2 of truth. Now we have a legal run Γ of the cirquent as an additional parameter, which was trivial (namely, $\Gamma = \langle\rangle$) in Definition 3.2 and hence not mentioned there.

Definition 4.3 Let C be a cirquent, * an interpretation, and Γ a legal run of C. In this context, with "true" to be read as "**true w.r.t.** $(^*, \Gamma)$", we say that:

- An L-port is true iff $L^* = \top$.
- A $\vee$-gate is true iff so is at least one of its children.
- A $\wedge$-gate is true iff so are all of its children.
- A resolved selectional gate is true iff so is its resolvent.
- No unresolved disjunctive selectional gate is true.
- Each unresolved conjunctive selectional gate is true.

Finally, we say that C is true iff so is its root.

As we just did in the above definition, when * and Γ are fixed in a context or are otherwise irrelevant, we may omit "w.r.t. $(^*, \Gamma)$" and just say "true".

Definition 4.4 Let C be a cirquent, * an interpretation, and Γ a legal run of C. Then Γ is a $\top$-won run of the game C^* iff C is true w.r.t. $(^*, \Gamma)$.

[6]The latter, of course, may not be the case if g is a choice gate, or a sequential gate with a finite outdegree.

Definitions 4.1 and 4.4, together, provide a definition of the game C^*, for any cirquent C and interpretation *. To such a game C^* we may refer as "the game represented by C under interpretation *", or as "C under interpretation *", or — when * is fixed or irrelevant — as "the game represented by C". We may also say that "* interprets C as C^*". Similarly for atoms and literals instead of cirquents. Also, in informal contexts we may identify a cirquent C or a literal L with a (the) game represented by it, and write C or L where, strictly speaking, C^* or L^* is meant. These and similar terminological conventions apply not only to the present section, but the rest of the paper as well.

We now need to generalize the definition of validity given in the previous section to cirquents in the sense of the present section. As it turns out, such a generalization can be made in two, equally natural, ways:

Definition 4.5 Let C be a cirquent in the sense of the present or any of the subsequent sections. We say that:

1. C is **weakly valid** iff, for any interpretation *, there is an HPM $\mathcal{M}$ such that $\mathcal{M}$ wins the game C^*.
2. C is **strongly valid** iff there is an HPM $\mathcal{M}$ such that, for any interpretation *, $\mathcal{M}$ wins the game C^*.

When $\mathcal{M}$ and C satisfy the second clause of the above definition, we say that $\mathcal{M}$ is a **uniform solution** for C. Intuitively, a uniform solution $\mathcal{M}$ for a cirquent C is an interpretation-independent winning strategy: since the "intended" or "actual" interpretation * is not visible to the machine, $\mathcal{M}$ has to play in some standard, uniform way that would be successful for any possible interpretation of C. To put it in other words, a uniform solution is a *purely logical solution*, which is based only on the *form* of a cirquent rather than any extra-logical *meaning* (interpretation) we have or could have associated with it.

It is obvious that for cirquents of the previous section, the weak and strong forms of validity coincide with what we simply called "validity" there. In general, however, not every weakly valid cirquent would also be strongly valid. A simplest example of such a weakly valid cirquent is $\neg p \sqcup p$.[7] Under any interpretation *, the game represented by this cirquent is won by one of the two HPMs $\mathcal{M}_1$ or $\mathcal{M}_2$, where $\mathcal{M}_1$ is the machine that selects the left disjunct and rests, while $\mathcal{M}_2$ is the machine that selects the right disjunct and rests. However, which of these two machines wins the game depends on whether * interprets p as $\bot$ or $\top$. In general, there is no *one* machine that wins the game for *any* possible interpretation. That is, the cirquent $\neg p \sqcup p$ has no uniform solution, and thus it is not strongly valid.

Which of the two versions of validity is more interesting depends on the motivational standpoint. Weak validity tells us what can be computed in principle. So, a computability-theoretician would focus on this concept. On the other hand, it is strong rather than weak validity that would be of interest in all application areas of CoL. There we want a logic on which a universal problem-solving machine can be based. Such a machine would or should be able to solve problems represented by logical expressions without any specific knowledge of the meanings of their atoms, i.e. without knowledge of the actual interpretation, which may vary from situation to situation or from application to application. Strong validity is exactly what fits the bill in this case. Throughout this paper, our primary focus will be on strong rather than weak validity.

To appreciate the difference between the parallel and choice groups of gates or connectives, let us compare the two cirquents of Figure 3.

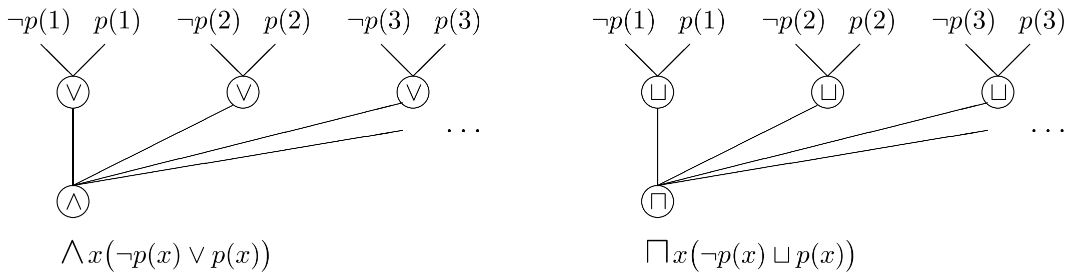

$\neg p(1)$ $p(1)$ $\neg p(2)$ $p(2)$ $\neg p(3)$ $p(3)$
$\vee$
$\wedge$
...
$\bigwedge x\big(\neg p(x) \vee p(x)\big)$
$\sqcup$
$\sqcap$
$\sqcap x\big(\neg p(x) \sqcup p(x)\big)$

Figure 3: Parallel versus choice gates

[7]If, however, we do not limit our considerations to perfect interpretations (see the footnote on page 7) and allow all interpretations in the definition of weak validity, this cirquent will no longer be weakly valid. In fact, for all well studied fragments of CoL, when interpretations are not required to be perfect, the weak and the strong versions of validity have been shown to yield the same classes of formulas. Strong validity in the CoL literature is usually referred to as *uniform validity*, and weak validity as (simply) *validity*.

The game represented by the left cirquent of Figure 3 is elementary, where no moves can or should be made by either player. It is also easy to see that this game (the only legal run $\langle\rangle$ of it, that is) is automatically won by $\top$, no matter what interpretation * is applied, so, it is both weakly and strongly valid. On the other hand, the game represented by the right cirquent of the same figure is not elementary. And it is neither strongly valid nor weakly valid. A legal move by $\bot$ in this game consists in selecting one of the infinitely many outgoing edges (and hence children) of the root, intuitively corresponding to choosing a value for x in the formula $\sqcap x\bigl(\neg p(x) \sqcup p(x)\bigr)$. And a(ny) legal move by $\top$ consists in selecting one of the two outgoing edges (and hence children) of one of the $\sqcup$-gates. Making more than one selection in the same choice (unlike toggling or sequential) gate is not allowed, so that a selection automatically also is the resolvent of the gate. The overall game is won by $\top$ iff either $\bot$ failed to make a selection in the root, or else, where i is the outgoing edge of the root selected by $\bot$ and a_i is the corresponding (ith, that is) child of the root, either (1) $\top$ has selected the left outgoing edge of a_i and $\neg p(i)$ is true, or (2) $\top$ has selected the right outgoing edge of a_i and $p(i)$ is true. There are no conditions on when the available moves should be made, and generally they can be made by either player at any time and in any order. So, in the present example, $\top$ can legally make selections in several or even all $\sqcup$-gates. But, of course, a reasonable strategy for $\top$ is to first wait till $\bot$ resolves the root (otherwise $\top$ wins), and then focus only on the resolvent of the root (what happens in the other $\sqcup$-gates no longer matters), trying to select the true child of it.

From the above explanation it should be clear that the right cirquent of Figure 3 expresses the problem of deciding (in the traditional sense) the predicate $p(x)$. That is, under any given interpretation *, the game represented by that cirquent has an algorithmic winning strategy by $\top$ if and only if the predicate $\bigl(p(x)\bigr)^*$ — which we simply write as $p(x)$ — is decidable. As not all predicates are decidable, the cirquent is not weakly valid, let alone being strongly valid.

To get a feel for sequential and toggling gates, let us look at Figure 4.

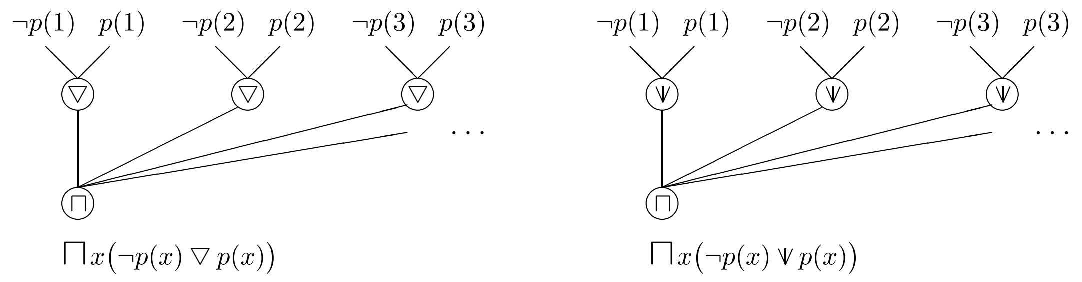

$\neg p(1)$ $p(1)$ $\neg p(2)$ $p(2)$ $\neg p(3)$ $p(3)$
$\bigtriangledown$
$\sqcap$
...
$\sqcap x\bigl(\neg p(x) \bigtriangledown p(x)\bigr)$
$\neg p(1)$ $p(1)$ $\neg p(2)$ $p(2)$ $\neg p(3)$ $p(3)$
⩛
$\sqcap$
...
$\sqcap x\bigl(\neg p(x)$ ⩛ $p(x)\bigr)$

Figure 4: Sequential versus toggling gates

The cirquents of Figure 4 look similar to the right cirquent of Figure 3. And the latter, as we know, represents the problem of *deciding* $p(x)$. Then what are the problems represented by the cirquents of Figure 4?

The left cirquent of Figure 4 represents the problem of *semideciding* $p(x)$. That is, under any given interpretation, the game represented by this cirquent is computable if and only if the predicate $p(x)$ is semidecidable (recursively enumerable). Indeed, suppose $p(x)$ is semidecidable. Then an algorithmic winning strategy for the game represented by the cirquent goes like this. Wait till the environment selects the ith child a_i of the root for some i. Then select the left child of a_i, after which start looking for a certificate of the truth of $p(i)$. If and when such a certificate is found, select the right child of a_i, and rest your case. It is obvious that this strategy indeed wins. For the opposite direction, suppose $\mathcal{M}$ is an HPM that wins the game represented by the cirquent. Then a semidecision procedure for the predicate $p(x)$ goes like this. After receiving an input i, simulate the work of $\mathcal{M}$ in the scenario where, at the beginning of the play, the environment selected the ith child a_i of the root. If and when you see in this simulation that $\mathcal{M}$ selected the right child of a_i, accept the input.

As for the right cirquent of Figure 4, it also represents a decision-style problem, which is further weaker than the problem of semideciding $p(x)$. This problem is known in the literature as *recursive approximation* (cf. [3], Definition 8.3.9). Recursively approximating $p(x)$ means telling whether $p(x)$ is true or not, but doing so in the same style as semideciding does in negative cases: by correctly saying "Yes" or "No" at some point (after perhaps taking back previous answers several times) and never reconsidering this answer afterwards. Observe that semideciding $p(x)$ can be seen as always saying "No" at the beginning and then, if this answer is incorrect, changing it to "Yes" at some later time; so, when the answer is negative, this will be expressed by saying "No" and never taking back this answer, yet without ever indicating that the answer is final and will not change.[8] Thus, the difference between semideciding and recursively approximating is that, unlike a semidecision procedure, a recursive approximation procedure can reconsider *both* negative and positive answers, and do so several times rather than only once. In perhaps more familiar terms, according to Shoenfield's Limit Lemma (Cf. [3], Lemma 8.3.12), a predicate $p(x)$ is recursively approximable (the problem of its recursive approximation has an algorithmic solution) iff $p(x)$ is of arithmetical complexity Δ_2, i.e., both $p(x)$ and its negation can be written in the form $\exists z \forall y\, s(z, y, x)$, where $s(z, y, x)$ is a decidable predicate.

We could go on and on illustrating how our formalism — even at the formula level — makes it possible to express various known and unknown natural and interesting properties, relations and operations on predicates or games as generalized predicates, but this can take us too far. Plenty of examples and discussions in that style can be found in, say, [9, 20, 21, 24]. Here we only want to point out the difference between our treatment of $\vee, \wedge$ (including quantifiers as infinite versions of $\vee, \wedge$) and the more traditional game-semantical approaches, most notably that of Hintikka's [4] game-theoretic semantics. The latter essentially treats $\vee, \wedge$ as we treat $\sqcup, \sqcap$ — namely, associates $\top$'s moves/choices with $\vee$ and $\bot$'s moves/choices with $\wedge$. Computability logic, on the other hand, in the style used by Blass [2] for the multiplicatives of linear logic, treats $A \vee B$ and $A \wedge B$ as parallel combinations of games: these are simultaneous plays on "two boards" (within the two components). In order to win $A \wedge B$, $\top$ needs to win in both components, while in order to win $A \vee B$, it is sufficient for $\top$ to win in just one component. No choice between A and B is expected to be made at any time by either player. Note that otherwise strong validity would not at all be an interesting concept: there would be no strongly valid cirquents except for some pathological cases such as the cirquent whose root is a childless conjunctive gate.

Another crucial difference between our approach and that of Hintikka, as well as the approach of Blass, is that we insist on the effectiveness of strategies while the latter allow any strategies. It is not hard to see that, if we allowed any (rather than only algorithmic) strategies, the system of our gates would semantically collapse to merely the parallel group. That is, a cirquent C (under whatever interpretation) would have a winning strategy by $\top$ if and only if C' does, where C' is the result of replacing in C every disjunctive selectional gate by $\vee$ and every conjunctive selectional gate by $\wedge$.

Anyway, an important issue for the present paper is that of the advantages of cirquents over formulas. As we remember, finite cirquents without selectional gates are more efficient tools of expression than formulas of classical logic are, but otherwise their expressive power is the same as that of formulas. How about finite cirquents in the more general sense of this section — ones containing selectional gates? In this case, finite (let alone infinite) cirquents are not only more efficient but also more expressive than formulas. To get some insights, let us look at Figure 5.

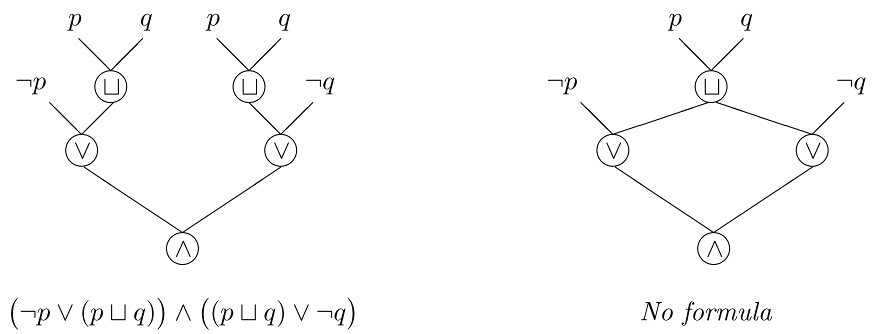


Figure 5: Unshared versus shared gates

On the left of Figure 5 we see a tree-like cirquent obtained from the non-tree-like cirquent on the right by duplicating and separating shared descendants in the same style as we obtained the left cirquent of Figure 1 from the right cirquent. The two cirquents of Figure 5, however, are not extensionally identical — after all, every legal run of the right cirquent contains at most one move while legal runs of the left cirquent may

[8]Unless, of course, the procedure halts by good luck. Halting without saying "Yes" can then be seen as an explicit indication that the original answer "No" was final.

contain two moves. The two cirquents are different in a much stronger sense than extensional non-identity though. A machine that selects the left outgoing edge of the left $\sqcup$-gate and the right outgoing edge of the right $\sqcup$-gate wins the game represented by the left cirquent of Figure 5 under any interpretation, so that the cirquent is strongly valid. On the other hand, the right cirquent of Figure 5 is obviously not strongly valid. Morally, the difference between the two cirquents is that, in the right cirquent, unlike the left cirquent, the subcirquent ("resource") $p \sqcup q$ is *shared* between the two $\wedge$-conjuncted parents. Sharing means that only one resolution can be made in the $\sqcup$-gate — a resolution that "works well" for both parents. There is no such resolution though, because the two parents "want" two different choices of a $\sqcup$-disjunct. In contrast, in the left cirquent, both of these conflicting "desires" can be satisfied as each parent has its own $\sqcup$-child, so that there is no need to coordinate resolutions.

In layman's terms, a shared resource (subcirquent) can be explained using the following metaphor. Imagine Victor and his wife Peggy own a joint bank account, with the balance of \$ 20,000, reserved for family needs. This resource can be seen as a shared choice combination of *truck*, *furniture*, and anything of family use that \$20,000 can buy.[9] However, if Victor prefers a truck while Peggy prefers furniture (and both cost \$20,000), then they will have to sit down and decide together whether furniture is more important for the family or a truck, as their budget does not allow to get both. The situation would be very different if both Victor and Peggy had their own accounts, each worth \$20,000. After all, the total balance in this case would be \$40,000 rather than \$20,000.

As we saw, the right cirquent of Figure 5 cannot be adequately translated into a formula using the standard way of turning non-trees into trees. However, using some non-standard and creative ways, a formula which is extensionally identical to that cirquent can still be found. For instance, one can easily see that $(p \sqcup q) \vee (\neg p \wedge \neg q)$ is such a formula (as long as its $\sqcup$-gate is given the same ID as the ID of the original cirquent's $\sqcup$-gate, of course). Well, we just got lucky in this particular case. "Luck" would not have been on our side if the cirquent under question was slightly more evolved, such as the one of Figure 6.

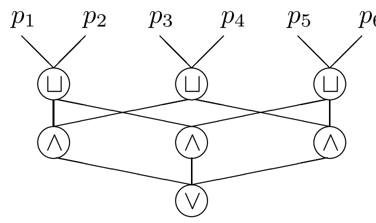


Figure 6: A more evolved example of sharing

5 Clustering selectional gates

We now further generalize **cirquents** by adding an extra parameter to them, called **clustering** (for selectional gates). The latter is nothing but a partition of the set of all selectional gates into subsets, called **clusters**, satisfying the condition that all gates within any given cluster have the same label (all are $\sqcup$-gates, or all are $\curlywedge$-gates, or ...) and the same outdegree. Due to this condition, we can talk about the **outdegree** of a cluster meaning the common outdegree of its elements, or the **type** of a cluster meaning the common type (label) of its elements. An additional condition that we require to be satisfied by all cirquents is that the question on whether any two given selectional gates are in the same cluster be decidable.

Just like nodes do, each cluster also has its own **ID**. For clarity, we assume that the ID of a cluster is the same as the smallest of the IDs of the elements of the cluster. The *extended ID* of a selectional gate is the expression n_k, where n is the ID of the gate and the subscript k is the ID of the cluster to which the gate belongs. When representing cirquents graphically, one could require to show the extended ID of each selectional gate and (just) the ID of any other node. More often, however, we draw cirquents in a lazy yet unambiguous way, where only cluster IDs are indicated; furthermore, such IDs can be (though not always will be) omitted for singleton clusters. Figure 7 shows the same cirquent, on the left represented fully and on the right represented in a lazy way, with the identical cluster ID "7" attached to two $\sqcup$-gates indicating

[9]If Victor and Peggy may change their mind several times, and the sellers' return policies are flexible enough, then this is a toggling combination rather than a choice one.

that they are in the same cluster, while the absence of a cluster ID for the middle $\sqcup$-gate indicating that it is in a different, singleton cluster (and so would be any other selectional gate if drawn without a cluster ID). Thus, altogether there are two clusters here, one — cluster 8 — containing gate 8, and the other — cluster 7 containing gates 7 and 9.

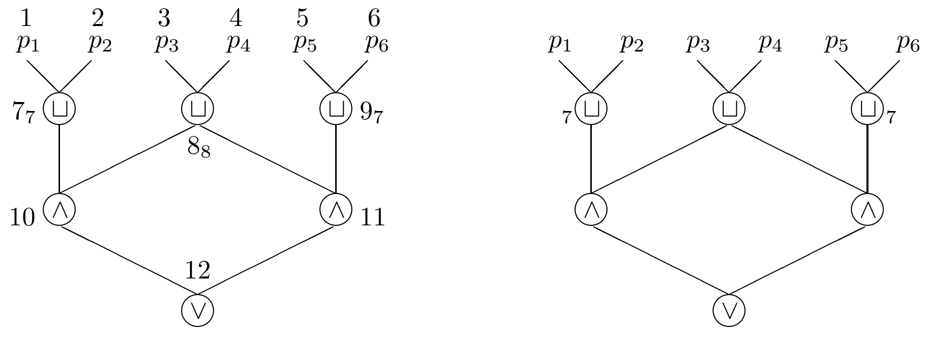


Figure 7: A cirquent with clustered selectional gates

It may be helpful for one's intuition to think of each cluster as a single gate-style physical device rather than a collection of individual gates. Namely, a cluster consisting of n gates of outdegree m, as a single device, would have n outputs and m n-tuples of inputs. Figure 8 depicts this new kind of a "gate" for the case when $n = 3$, $m = 2$ and the type of the cluster is $\sqcup$.

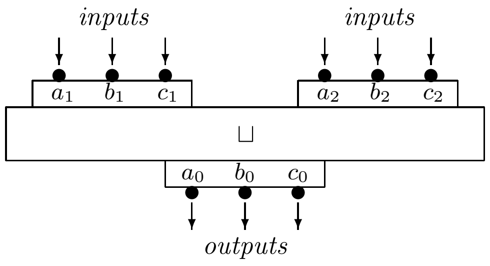


Figure 8: Clusters as generalized gates

The device shown in Figure 8 should be thought of as a switch that can be set to one of the positions 1 or 2 (otherwise no signals will pass through it). Setting it to 1 simultaneously connects the three input lines a_1, b_1 and c_1 to the output lines a_0, b_0 and c_0, respectively (the three lines are parallel, isolated from each other, so that no signal can jump from one line to another). Similarly, setting the switch to 2 connects a_2, b_2 and c_2 to a_0, b_0 and c_0, respectively. This is thus an "*either* (a_1, b_1, c_1) *or* (a_2, b_2, c_2)" kind of a switch; combinations such as, say, (a_1, b_2, c_1), are not available.

Figure 9 shows the cirquent of Figure 7 with its clusters re-drawn in the style of Figure 8.

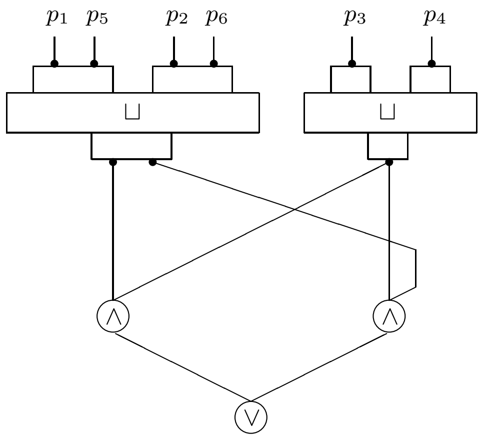


Figure 9: An alternative representation of the cirquent of Figure 7

Representing clusters in the way we have just done illustrates that they are nothing but generalized gates. In fact, this is a very natural generalization. Namely, a gate in the ordinary sense is the special case of this general sort of a gate where the number of output lines (as well as input lines in each group of inputs) equals 1. Technically, however, we prefer to continue seeing clusters as sets of ordinary gates as they were officially defined at the beginning of this section. So, drawings in the style of Figures 8 or 9 will never reemerge in this paper, and whatever was said about clusters as individual "gates" of a new type can be safely forgotten.

Cirquents of the previous section should be viewed as special cases of cirquents in the new sense of this section. Namely, they are the cases where each selectional gate forms its own, single-element cluster. With this view, the semantical concepts we reintroduce in this section conservatively generalize those of the previous section.

The definition of a legal run of the game represented by a cirquent C is the same as before (Definition 4.1), with the difference that now moves are made within clusters rather than individual gates. That is, each move of a legal run of C looks like $c.i$, where c is the ID of a cluster (rather than gate), and i is a positive integer not exceeding the outdegree of that cluster. The intuitive meaning of such a move $c.i$ is selecting the ith outgoing edge (together with the corresponding child) simultaneously in *each* gate belonging to cluster c. All other conditions on the legality of runs remain literally the same as before. Anyway, let us not be lazy to fully (re)produce such a definition:

Definition 5.1 Let C be a cirquent, * an interpretation, and Φ a position. Φ is a **legal position** of the game C^* iff, with a "cluster" below meaning a cluster of C, the following conditions are satisfied:

1. Each labmove of Φ has one of the following forms:

   (a) $\top c.i$, where c is a $\curlyvee$-, $\triangledown$- or $\sqcup$-cluster and i is a positive integer not exceeding the outdegree of c;

   (b) $\bot c.i$, where c is a $\curlywedge$-, $\vartriangle$- or $\sqcap$-cluster and i is a positive integer not exceeding the outdegree of c.

2. Whenever c is a choice cluster, Φ contains at most one occurrence of a labmove of the form $\wp c.i$.

3. Whenever c is a sequential cluster and Φ is of the form $\langle \ldots, \wp c.i, \ldots, \wp c.j, \ldots \rangle$, we have $i < j$.

So, for instance, $\langle \top 8.1, \top 7.2 \rangle$ is a legal run of the cirquent of Figure 7, but $\langle \top 8.1, \top 9.2 \rangle$ is not, because 9 is (a gate ID but) not a cluster ID. To summarize once again, selections (moves) in individual *gates* are no longer available. Rather, they should be made in *clusters*.

Definition 5.2 In the context of a given cirquent C and a legal run Γ of C, we will say that a selectional gate g of a cluster c is **unresolved** iff either no moves of the form $c.j$ have been made in Γ, or infinitely many such moves have been made. Otherwise g is **resolved** and, where $c.i$ is the last move of the form $c.j$ made in Γ, the child pointed at by the ith outgoing edge of g is said to be the **resolvent** of g.

With the terms "unresolved", "resolved" and "resolvent" conservatively redefined this way, the definition of **truth** for a cirquent and its nodes is *literally* the same[10] in our present case as in the case of the cirquents of the previous section (Definition 4.3), so we do not reproduce it here. The same applies to the definition of the **Wn** components of the games represented by cirquents (Definition 4.4).

Let us look at the game represented by the cirquents of Figure 7 once again. The meaning of the move "7.2" in this game is selecting outgoing edge #2 in *both* gates of cluster 7. Intuitively, the effect of such a move is connecting gate 10 directly to node 2 and gate 11 directly to node 6. Thus, the move "7.2" can be seen as a choice (choice #2) *shared* between gates 10 and 11; that shared choice, however, yields different, unshared results for the two gates: result 2 for gate 10 while result 6 for gate 11. This sort of sharing is very different from the sort of sharing represented by gate 8: the effect of the move "8.1" is sharing both the choice #1 as well as the result 3 of that choice.

Back to the world of Victor and Peggy, imagine they are in their family car on a road between two cities A and B. Victor likes sports but never goes to theaters. Peggy likes theaters but never attends games. There is a basketball game and a ballet show tonight in city A. And there is a football game and an opera show in city B. The shared choice/move in this situation is a choice between "*drive to A*" and "*drive to B*" (they only have one car!). The outcomes of either choice, however, are not shared. For instance, the outcome of the choice "*drive to A*" is "*see the basketball match*" for Victor while "*see the ballet*" for Peggy. Victor and Peggy can negotiate and decide between the two pairs (*Basketball*, *Ballet*) or (*Football*, *Opera*). But the pairs (*Basketball*, *Opera*) and (*Football*, *Ballet*) are not available.

As for the stronger type of sharing corresponding to the middle $\sqcup$-gate of Figure 7, it can be explained by the following modification of the situation: Both Victor and Peggy are fond of Impressionism. There is a Monet exhibition in city A, and a Pissarro exhibition in city B. The action/choice — driving to A or driving to B — is again shared. But now so is the corresponding outcome "*see Monet's paintings*" or "*see Pissarro's paintings*".

While we could continue elaborating on independent philosophical and mathematical/technical motivations for introducing clustering, here we just want to point out the very direct connections of our approach to the already well motivated *IF* (*independence-friendly*) *logic*. We assume that the reader is familiar with the basics of the latter, or else he or she may take a quick look at the concise yet complete (for our purposes) overview of the subject given in [32]. It should be noted that we use the term "IF logic" in a generic sense, making no terminological distinction between Hintikka and Sandu's [5] original version of IF logic and Hodge's [6] generalization of it termed in [7] *slash logic*. In fact, both the syntax and the semantics for IF formulas that we use are those of slash logic. It is assumed that all formulas of IF logic are written in negation normal form, and that different occurrences of quantifiers in them always bind different variables. We also assume that only existential quantifiers and disjunctions are slashed in the formulas of IF logic — it is known that, in IF logic (as opposed to what is called *extended IF logic*), slashing universal quantifiers or conjunctions is redundant, having no effect on the semantical meanings of formulas. Finally, without much (if any) loss of generality, we assume that the semantics of IF logic exclusively deals with models with countable domains; namely, such a domain is always $\{1, 2, 3, \ldots\}$ or some nonempty finite initial portion of it.

Computability logic insists on algorithmicity of $\top$'s strategies, while IF logic, in its game semantics, allows any strategies. Let us, for now, consider the version of IF logic which differs from its canonical version only in that it, like CoL, requires $\top$'s (which there is usually called $\exists$-Player, or Verifier, or Eloise) strategies to be effective, while imposing no restrictions on $\bot$'s ($\forall$-Player's, Falsifier's, Abelard's) strategies. Call this version of IF logic **effective IF logic**.

Remember that the outgoing edges of each node of a cirquent come in a fixed left-to-right order: edge #1, edge #2, edge #3, etc. Let us call these numbers 1, 2, 3, etc. the **order numbers** of the corresponding edges.

We now claim that the fragment of our cirquent logic where cirquents are allowed to have only two — $\wedge$ and $\sqcup$ — sorts of gates is sufficient to cover effective IF logic, and far beyond. A verification of this claim — perhaps at the philosophical rather than technical level — is left to the reader.

Namely, each formula E of effective IF logic can be understood as the cirquent obtained from it through performing the following operations:

Description 5.3

[10]Unlike Definition 5.1 which, at least, changed the word "gate" to the word "cluster" when reproducing the corresponding Definition 4.1.

1. Ignoring slashes, write E in the form of a tree-like cirquent, understanding $\forall$ as a "long" $\wedge$-conjunction and $\exists$ as a "long" $\vee$-disjunction. This way, each occurrence O of a quantifier, conjunction or disjunction gives rise to one (if O is not in the scope of a quantifier) or many (if O is in the scope of a quantifier) gates. We say that each such gate, as well as each outgoing edge of it, **originates** from O. Also, since we assume that different occurrences of quantifiers in IF formulas always bind different variables, instead of saying "…originates from the occurrence of $\forall y$ (or $\exists y$)" we can unambiguously simply say "…originates from y".

2. Change the label of each $\vee$-gate to $\sqcup$.

3. Cluster the disjunctive gates so that, any two such gates a and b belong to the same cluster if and only if they originate from the same occurrence of $\exists x/y_1,\ldots,y_n$ or $\vee/y_1,\ldots,y_n$ in E, and the following condition is satisfied:
    - Let $e^a_1,\ldots,e^a_k$ and $e^b_1,\ldots,e^b_k$ be the paths (sequences of edges) from the root of the tree to a and b, respectively. Then, for any $i\in\{1,\ldots,k\}$, the order numbers of the edges e^a_i and e^b_i are identical unless these edges originate from one of the variables $y_1,\ldots,y_n$.

Let us see an example to visualize how our construction works. A traditional starting point of introductory or survey papers on IF logic is the formula

$$\forall x\exists y\forall z\exists t/x,y\ p(x,y,z,t). \tag{1}$$

Technically, its meaning can be expressed by the second-order formula $\exists f\exists g\forall x\forall z\ p\big(x,f(x),z,g(z)\big)$[11] or the following formula with Henkin's branching quantifiers:

$$\begin{matrix}\forall x\exists y\\ \forall z\exists t\end{matrix}\ p(x,y,z,t),$$

with the shape of the quantifier array indicating that the two quantifier blocks $\forall x\exists y$ and $\forall z\exists t$ are independent of each other even though, in (1), one occurs in the scope of the other. We agreed earlier to write $\bigwedge,\bigvee$ instead of $\forall,\exists$. So, we rewrite (1) as $\bigwedge x\bigvee y\bigwedge z\bigvee t/x,y\ p(x,y,z,t)$. This formula, however, is not yet adequate. The philosophy of IF logic associates the intuitions of *finding* (rather than just *existence*) with existential quantifiers or disjunctions; and, as we know, it is the operator/gate $\sqcup$ whose precise meaning is actually *finding* things (rather than $\vee$, which is merely about *existence* of things). So, $\bigvee$ should be replaced with $\bigsqcup$, and we get the formula $\bigwedge x\bigsqcup y\bigwedge z\bigsqcup t/x,y\,p(x,y,z,t)$ which, ignoring the slash for now, can be seen as a tree-like cirquent in the sense of the previous section. It now remains to account for the slash by adequately clustering the cirquent. Namely, the clustering should make sure that, whenever a and b are two upper-level (those originating from t) $\sqcup$-gates such that the paths from the root to them — seen not as sequences of edges but as sequences of the corresponding order numbers — only differ in their first two elements (the ones originating from x and y, of which t should be independent), a and b are in the same cluster. Figure 10 illustrates this arrangement. For compactness considerations, in this figure we have assumed that the universe of discourse (the set over which x,y,z,t range) is just $\{1,2\}$; also, we have written p_{1111}, p_{1112}, etc. instead of $p(1,1,1,1)$, $p(1,1,1,2)$, etc.

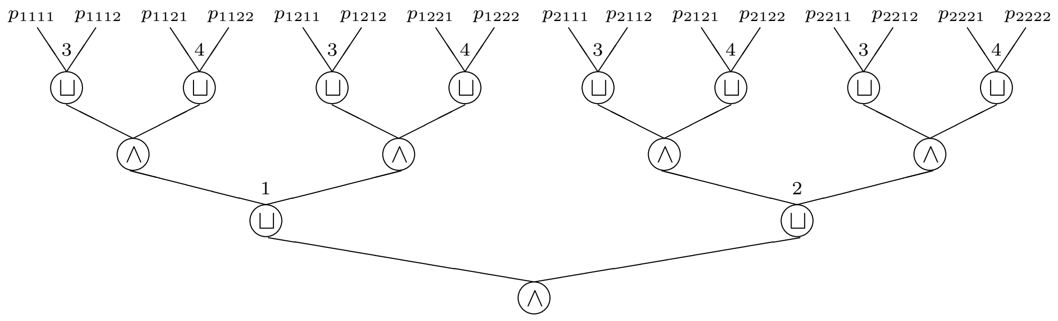

p_{1111} p_{1112} p_{1121} p_{1122} p_{1211} p_{1212} p_{1221} p_{1222} p_{2111} p_{2112} p_{2121} p_{2122} p_{2211} p_{2212} p_{2221} p_{2222}
3
4
3
4
3
4
3
4
1
2

Figure 10: Mimicking $\forall x\exists y\forall z\exists t/x,y\ p(x,y,z,t)$ (when the universe is $\{1,2\}$)

[11]As long as we deal with effective IF logic, one should require here that f and g range over recursive functions.

To feel the difference created by clustering, let us consider the interpretation that sends the four atoms p_{1111}, p_{1122}, p_{2212}, p_{2221} to $\top$ and sends all other atoms to $\bot$. Then the game represented by the cirquent of Figure 10 cannot be won (by $\top$). On the other hand, it would be winnable if there was no clustering. Further, it is also winnable if clustering is made finer (yet not trivial) as done in the cirquent of Figure 11. The latter expresses the "slightly" modified form $\forall x \exists y \forall z \exists t/x\ p(x,y,z,t)$ of (1), and is won by the HPM that generates the run

$$\langle \top 1.1,\ \top 3.1,\ \top 4.2,\ \top 2.2,\ \top 5.2,\ \top 6.1 \rangle$$

(or any permutation of the above). Note that the same run is simply not a legal run of the cirquent of Figure 10.

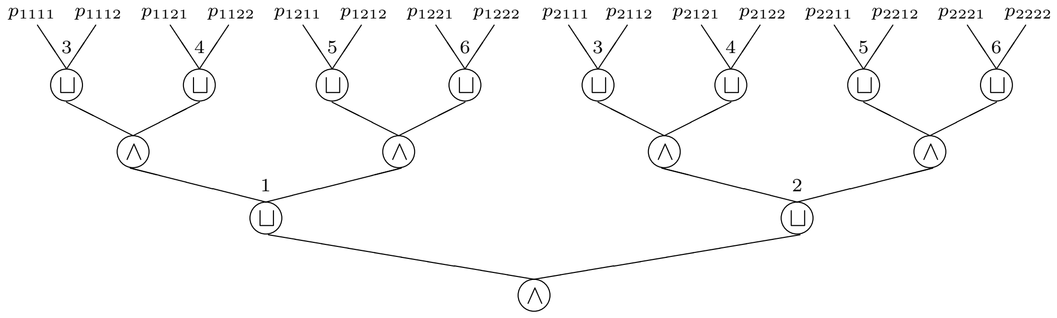

p_{1111} p_{1112} p_{1121} p_{1122} p_{1211} p_{1212} p_{1221} p_{1222} p_{2111} p_{2112} p_{2121} p_{2122} p_{2211} p_{2212} p_{2221} p_{2222}
3 4 5 6 3 4 5 6
1 2

Figure 11: Mimicking $\forall x \exists y \forall z \exists t/x\ p(x,y,z,t)$ (when the universe is $\{1,2\}$)

While $(\wedge, \sqcup)$-cirquents allow us to fully capture effective IF logic, they, at the same time, are significantly more expressive than the latter is, even if — or, especially if — we limit ourselves to finite cirquents, and correspondingly limit IF formulas to propositional ones. As mentioned earlier, IF logic has no smooth approach at the purely propositional level, and is forced to severely limit the forms of (meaningful) propositional-level formulas as, for instance, done in [30]. In particular, it encounters serious difficulties in allowing independence from conjunctions or disjunctions (rather than quantifiers). These problems are automatically neutralized under our approach. We do not introduce any restrictions whatsoever on the forms of cirquents or allowable clusterings in them; yet, all such expressions are semantically meaningful.

Of course, a penalty for the higher expressive power of cirquents is the awkwardness associated with the necessity to draw cirquents instead of writing formulas. But, again, various syntactic shortcuts can be introduced to make life easier, with recurrence operators, quantifiers or the slash notation being among such shortcuts. It should also be remembered that drawing cirquents may be some annoyance for humans (when writing papers) but not for computers (when using logic in their work); the latter, in fact, would *much* prefer cirquents, as they are exponentially more efficient means of expression than formulas are. In any case, a more significant achievement than expressiveness is probably avoiding the necessity to deal with the unplayable and troublemaking imperfect-information games on which the traditional semantics for IF logic are based. We owe this effect to the fact that clustering enforces at the *game* level what IF logic calls *uniformity* and tries to enforce at the *strategy* level. Rather than relying on players' integrity in expecting that they — to their disadvantage — will conscientiously forget the moves they have already seen but of which their actions should be independent, clustering simply makes "cheating" physically impossible.

But it should be remembered that "effective IF logic" is just our present invention which, while both mathematically and philosophically reasonable, is not at all the same as IF logic in its canonical form. So, a true merger between CoL and IF logic would seemingly require a compromise from one of them: either IF logic should adopt the requirement of effectiveness of strategies, or computability logic should drop this (central to its philosophy) requirement. Probably neither camp would be willing to make a compromise.

Fortunately, there is no real need for any compromises. The following section further generalizes the concept of cirquents in a conservative way. The new cirquents, unlike the cirquents of the present section, are powerful enough to express anything that the traditional ("non-effective") IF logic can. This is achieved through extending the idea of clustering from selectional gates to $\vee$-gates, yet without associating any moves with such gates or clusters.

6 Clustering $\vee$-gates

A **cirquent** in the sense of the present section is the same as one in the sense of the previous section, with the difference that now not only the set of selectional gates is partitioned into clusters, but also the set of $\vee$-gates (but not the set of $\wedge$-gates — not yet, at least). The condition on clustering is the same as before: all gates within a given cluster are required to have the same type (label) and same outdegree. Cirquents in the sense of the previous section are seen as special cases of cirquents in the present sense, namely, the cases where all $\vee$-clusters are singletons.

The **legal positions** of the game represented by a cirquent in this new sense are defined in literally the same way as before (Definition 5.1). So, clustering $\vee$-gates in a cirquent does not affect the set of its legal runs.

By a **metaselection** for a cirquent C we will mean a (not necessarily effective) partial function f from $\vee$-clusters of C to the set of positive integers, such that, for any $\vee$-cluster c, whenever defined, $f(c)$ does not exceed the outdegree of c.[12]

In the context of a given cirquent C, a legal run Γ of C and a metaselection f for C, we will say that a $\vee$-gate g of a cluster c is **unresolved** iff f is undefined at c (note that a childless $\vee$-gate will always be unresolved). Otherwise g is **resolved**, and the child of it pointed at by the $f(c)$th outgoing edge is said to be the **resolvent** of g. As for the same-name concepts "unresolved", "resolved" and "resolvent" for selectional gates, they are defined literally as before (Definition 5.2). Note that these three concepts depend on f but not Γ when g is a $\vee$-gate, while they depend on Γ but not f when g is a selectional gate. The function f thus acts as a "metaextension" of Γ. Intuitively, it can be thought of as selections in $\vee$-clusters made by the guardian angel of $\top$ in favor of $\top$ *after* the actual play took place (rather than *during* it), even if the latter lasted infinitely long; unlike $\top$, its guardian angel has magic — nonalgorithmic — intellectual powers to make best possible selections. Technically, however, selections by the "angel", unlike selections made by either player, are not moves of the game.

The following definition refines the earlier definitions of truth by relativizing this concept — renamed into *metatruth* — with respect to a metaselection as an additional parameter.

Definition 6.1 Let C be a cirquent, * an interpretation, Γ a legal run of C, and f a metaselection for C. In this context, with "metatrue" to be read as "**metatrue w.r.t.** $(^*, \Gamma, f)$", we say that:

- An L-port is metatrue iff $L^* = \top$.
- A resolved selectional gate is metatrue iff so is its resolvent.
- No unresolved disjunctive selectional gate is metatrue.
- Each unresolved conjunctive selectional gate is metatrue.
- A $\vee$-gate is metatrue iff it is resolved and its resolvent is metatrue.
- A $\wedge$-gate is metatrue iff so are all of its children.

Finally, we say that C is metatrue iff so is its root.

The following definition brings us from metatruth back to truth.

Definition 6.2 Let C be a cirquent, * an interpretation, and Γ a legal run of C. We say that C is **true** w.r.t. $(^*, \Gamma)$ iff there is a metaselection f for C such that C is metatrue w.r.t. $(^*, \Gamma, f)$.

It is left to the reader to see why the new concept of truth is a conservative generalization of its earlier counterparts. The same applies to the following definition, which completes our definition of the game C^* represented by any given cirquent C under any given interpretation *.

Definition 6.3 Let C be a cirquent, * an interpretation, and Γ a legal run of C. Then Γ is a $\top$-won run of the game C^* iff C is true w.r.t. $(^*, \Gamma)$.

[12] An equivalent approach would be letting f be a *total* function from the set of $\vee$-clusters to the set $\{0, 1, 2, 3, \ldots\}$ of *natural numbers* (rather than positive integers); then, instead of saying that f is undefined at c, we could simply say that $f(c) = 0$.

We now claim that any formula E of (this time ordinary, "non-effective") IF logic can be adequately written as a tree-like cirquent C with only $\vee$ and $\wedge$ gates; namely, such a cirquent C is obtained from E through applying the steps 1 and 3 of Description 5.3, with step 2 omitted. For any interpretation *, we will then have $C^* = \top$ iff E is true (under the same interpretation of atoms) in IF logic. Figure 12 shows an example. As an easy exercise, the reader may want to verify that the cirquent of that figure is $\bot$ under the interpretation which sends p_{1111}, p_{1122}, p_{2212}, p_{2221} to $\top$ and sends all other atoms to $\bot$. Note that this cirquent represents an elementary game, unlike its counterpart from Figure 10.

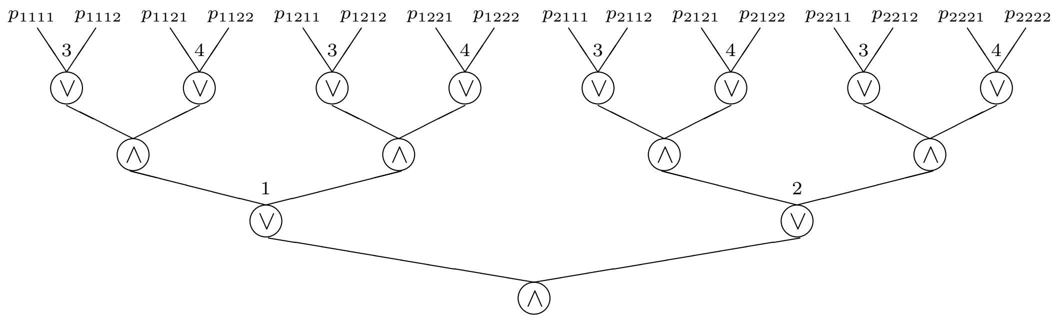


Figure 12: Representing $\forall x \exists y \forall z \exists t/x,y \; p(x, y, z, t)$ (when the universe is $\{1, 2\}$)

Just as for any other claims made in this paper regarding connections to IF logic, we are not attempting to provide a proof of our present claim. Such a proof would require reproducing and analyzing one of the semantics accepted/recognized in the IF logic community, which could take us too far — the present paper is on computability logic rather than IF logic after all and, even if a known semantics of IF logic and the corresponding fragment of the semantics of CoL turned out to be not exactly equivalent, a question would arise about which one is a more adequate materialization of the original philosophy of IF logic.

7 Clustering all gates; ranking

Other than the claimed fact that the cirquents of the previous section achieve the full expressive power of IF logic, there are no good reasons to stop at those cirquents. Indeed, if we clustered selectional and $\vee$-gates, why not do the same with the remaining $\wedge$ type of gates? Naturally, the semantics of clustered $\wedge$ gates would have to be symmetric to that of clustered $\vee$-gates. Namely, a universally quantified metaselection h should be associated with them, as we associated an existentially quantified metaselection f with $\vee$-clusters in the previous section. Such an h can be thought of as a guardian angel of $\bot$ that makes selections in $\wedge$-clusters in favor of $\bot$ after the game has been played by the two players. One can show that then, no matter how the $\wedge$-gates are clustered, truth in the sense of the previous section is equivalent to an assertion that sounds like the right side of Definition 6.2 but starts with the words "*there is an f such that, for all h,...*" instead of just "*there is an f such that...*". But then the question comes: why this quantification order and not "*for all h there is an f such that...*", which would obviously yield a different yet meaningful concept of truth?[13] Again, there is no good answer, and here we see the need for a yet more general approach that would be flexible enough to handle the semantical concepts induced by either quantification order. This brings us to the idea of introducing one more parameter into cirquents, which we call *ranking*. The latter is an indication of in what order selections by the "guardian angels" should be made. Furthermore, we allow not just one but several "guardian angels" for either player, with each "angel" responsible for a certain subset of clusters rather than all clusters of a given type. Then, again, ranking fixes the order in which these multiple "angels" should make their selections. Let us get down to formal definitions to make these intuitions precise and more clear.

A **cirquent** in the sense of the present section is the same as one in the sense of the previous section, with the following two changes. Firstly, now the set of *all gates* (including $\wedge$-gates) is partitioned into clusters, with each cluster, as before, satisfying the condition that all gates in it have the same type and the same

[13]This predicate of truth, in contrast to the previous one, would depend on how $\wedge$-gates are clustered but not on how $\vee$-gates are clustered.

outdegree. Secondly, there is an additional parameter called **ranking**. The latter is a partition of the set of all parallel ($\vee$ and $\wedge$) clusters into a finite number of subsets, called **ranks**, arranged in a linear order, with each rank satisfying the condition that all clusters in it have the same type (but not necessarily the same outdegree). A rank containing $\wedge$-clusters is said to be **conjunctive**, and a rank containing $\vee$-clusters **disjunctive**. Since the ranks are linearly ordered, we can refer to them as the 1st rank, the 2nd rank, etc. or rank 1, rank 2, etc. Also, instead of "cluster c is in the ith rank", we may say "c is of (or has) rank i".

Cirquents in the sense of the previous section are understood as special cases of cirquents in the present sense, namely, as cirquents where all $\wedge$-clusters are singletons of the highest rank, and all $\vee$-clusters (which are not necessarily singletons) are of the lowest rank. Here we say "highest rank" and "lowest rank" instead of "rank 2" and "rank 1" just for safety, because, if one of the two types of parallel gates is absent, then rank 1 would be both highest and lowest; and, if there are no parallel gates at all, the cirquent would not even have rank 1.

Let C be a cirquent with k ranks, and let $1 \leq i \leq k$. An ***i*-metaselection** for C is a (not necessarily effective) partial function from the ith rank of C to the set of positive integers, satisfying the condition that, for any given cluster c of the ith rank, whenever $f_i(c)$ is defined, its value does not exceed the outdegree of c. And a (simply) **metaselection** for C is a k-tuple $(f_1, \ldots, f_k)$ where, for each $1 \leq i \leq k$, f_i is an i-metaselection for C.

Clustering parallel gates and ranking has no effect on the set of **legal runs** of the game represented by a cirquent, so the definition of legal positions for cirquents of Section 5 (Definition 5.1) transfers to the present case without any changes.

Definition 7.1 In the context of a given cirquent C with k ranks, a legal run Γ of C and a metaselection $\vec{f} = (f_1, \ldots, f_k)$ for C, we will say that a $\vee$-gate g of a cluster c is **unresolved** iff, where i is the rank of c, the function f_i is undefined at c. Otherwise g is **resolved**, and the child of it pointed at by the $f_i(c)$th outgoing edge is said to be the **resolvent** of g. As for the same-name concepts "unresolved", "resolved" and "resolvent" for selectional gates, they are defined literally as before (Definition 5.2).

The following definition of metatruth can be seen to conservatively generalize its predecessor, Definition 6.1:

Definition 7.2 Let C be a cirquent, * an interpretation, Γ a legal run of C, and $\vec{f}$ a metaselection for C. In this context, with "metatrue" to be read as "**metatrue w.r.t.** $(^*, \Gamma, \vec{f})$", we say that:

- An L-port is metatrue iff $L^* = \top$.
- A resolved gate (of any of the eight types) is metatrue iff so is its resolvent.
- No unresolved disjunctive gate (of any of the four types) is metatrue.
- Every unresolved conjunctive gate (of any of the four types) is metatrue.

Finally, we say that C is metatrue iff so is its root.

The following definition brings us from metatruth back to truth. Again, it can be seen to conservatively generalize its predecessor, Definition 6.2:

Definition 7.3 Let C be a cirquent with k ranks, * an interpretation, and Γ a legal run of C. We say that C is **true** w.r.t. $(^*, \Gamma)$ iff

$$\mathcal{Q}_1 f_1 \ldots \mathcal{Q}_k f_k \textit{ such that } C \textit{ is metatrue w.r.t. } (^*, \Gamma, (f_1, \ldots, f_k)).$$

Here each $\mathcal{Q}_i f_i$ abbreviates the phrase "for every i-metaselection f_i for C" if the ith rank is conjunctive, and "there is an i-metaselection f_i for C" if the ith rank is disjunctive.

With truth redefined this way, the (remaining) **Wn** component of the game C^* represented by a cirquent C under an interpretation * is defined as before. Namely, a legal run Γ of C^* is considered $\top$-won iff C is true w.r.t. $(^*, \Gamma)$.

To understand what we have achieved by introducing ranking and why such a generalization of cirquents was naturally called for, let us, for simplicity, limit our attention to selectional-gate-free cirquents. This fragment of our logic can be seen to be sufficient to capture and naturally generalize the conservative extension of IF logic known as *extended IF logic* (cf. [32]). The latter, in addition to what IF logic calls *strong negation* $\sim$, also considers *weak negation* $\neg$. While $\sim E$ simply means the result of changing in E each operator and atom to its dual (p to $\neg p$ and vice versa, $\forall$ to $\exists$ and vice versa, $\wedge$ to $\vee$ and vice versa) and hence there is no real need to have $\sim$ as a primitive, weak negation $\neg$ in (extended) IF logic is quite problematic. Namely, the latter does not act like an ordinary operator that can be applied anywhere in a formula; rather, extended IF logic (essentially) only allows $\neg$ to be applied to entire IF formulas, thus deeming meaningless and illegal expressions such as, say, $\exists u \neg \forall x \exists y \forall z \exists t/x\, p(x, y, z, t, u)$. This is so because the traditional approaches to IF logic are not general enough to directly provide a semantics for $\neg$. This odd situation makes it evident that more general approaches are necessary. Our approach can claim to be one that fits the bill.

As noted earlier, the reader is expected to be familiar with the basic concepts and ideas of IF logic and, specifically, the two concepts of negation that we are discussing. So, we only explain the present point through particular examples.

Our earlier assumption was that only existential quantifiers and disjunctions were slashed in formulas of non-extended IF logic, as slashing universal quantifiers or conjunctions is generally redundant there. The same is no longer the case if one deals with negations though. Accordingly, from now on, we depart from the above assumption and allow slashing any operators in what we consider to be legitimate formulas of IF logic.

Earlier we saw how to translate an IF formula E into an equivalent cirquent. Now we conservatively refine that translation method in a way that accounts for the possibility that E contains slashed conjunctions and/or universal quantifiers. Namely, E should be understood as the cirquent E° defined below:

Description 7.4 Let E be any formula of (non-extended) IF logic. We define E° as the cirquent obtained from E through performing the following steps:

1. Ignoring slashes, write E in the form of a tree-like cirquent, understanding $\forall$ as a "long" $\wedge$-conjunction and $\exists$ as a "long" $\vee$-disjunction. This way, each occurrence O of a quantifier, conjunction or disjunction gives rise to one (if O is not in the scope of a quantifier) or many (if O is in the scope of a quantifier) gates. We say that each such gate, as well as each outgoing edge of it, **originates** from O. Also, since we assume that different occurrences of quantifiers in IF formulas always bind different variables, instead of saying "…originates from the occurrence of $\forall y$ (or $\exists y$)" we can unambiguously simply say "…originates from y".

2. Cluster the gates so that, any two gates a and b belong to the same cluster if and only if they originate from the same occurrence of $\exists x/y_1, \ldots, y_n$, $\vee/y_1, \ldots, y_n$, $\forall x/y_1, \ldots, y_n$ or $\wedge/y_1, \ldots, y_n$ in E, and the following condition is satisfied:

   - Let $e_1^a, \ldots, e_k^a$ and $e_1^b, \ldots, e_k^b$ be the paths (sequences of edges) from the root of the tree to a and b, respectively. Then, for any $i \in \{1, \ldots, k\}$, the order numbers of the edges e_i^a and e_i^b are identical unless these edges originate from one of the variables $y_1, \ldots, y_n$.

3. Impose ranking on the resulting cirquent, putting all $\vee$-clusters into the lowest rank and all $\wedge$-clusters into the highest rank.

We claim that such a translation of IF formulas E to cirquents E° is adequate.

We further claim that any formula $\neg E$ of extended IF logic adequately translates into the cirquent $(\neg E)^\circ$ defined below:

Description 7.5 Let E be any formula of (non-extended) IF logic. We define $(\neg E)^\circ$ as the cirquent obtained from E through performing the following steps:

1. Turn E into E° according to Description 7.4.

2. Change in E° every port label (literal) p to $\neg p$ and vice versa, and also change every gate label $\vee$ to $\wedge$ and vice versa.

Finally, we claim that any formula $\sim E$ of IF logic adequately translates into the cirquent $(\sim E)^\circ$ defined below:

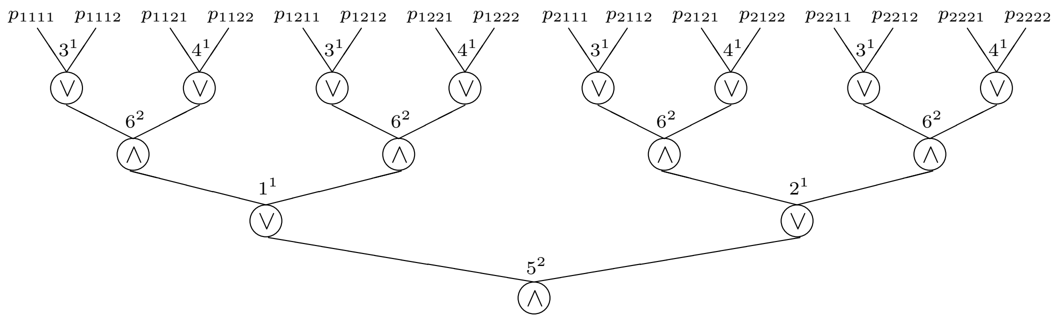
Description 7.6 Let E be any formula of (non-extended) IF logic. We define $(\sim E)^\circ$ as the cirquent obtained from E through performing the following steps:

1. Turn E into $(\neg E)^\circ$ according to Description 7.5.
2. Swap the ranks in the resulting cirquent. That is, make all elements of the formerly lowest rank now belong to the highest rank, and vice versa.

Figures 13, 14 and 15 illustrate applications of our translations to the IF-logic formula

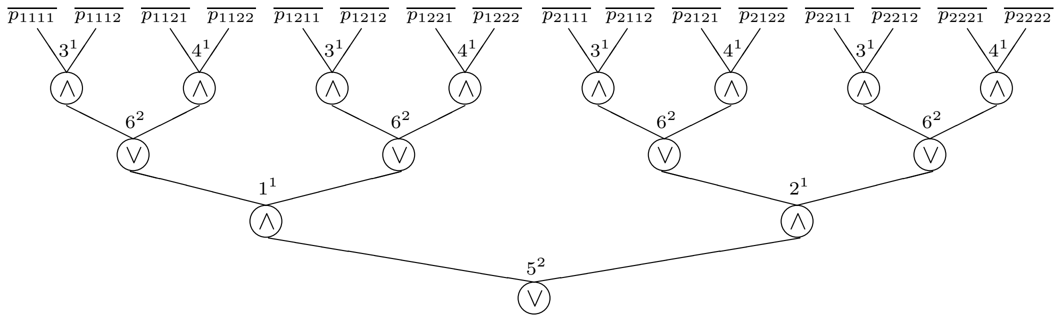
$$\forall x \exists y \forall z/x,y \exists t/x,y\ p(x,y,z,t)$$

and two forms of its negation. As before, the universe of discourse here is assumed to be $\{1,2\}$. For compactness considerations, we have written $\overline{p_{1111}}$, $\overline{p_{1112}}$, etc. instead of $\neg p_{1111}$, $\neg p_{1112}$, etc. To each gate in those figures we have attached an expression of the form n^m. It should be understood as an indication that the gate belongs to cluster n, and that such a cluster n is of rank m.



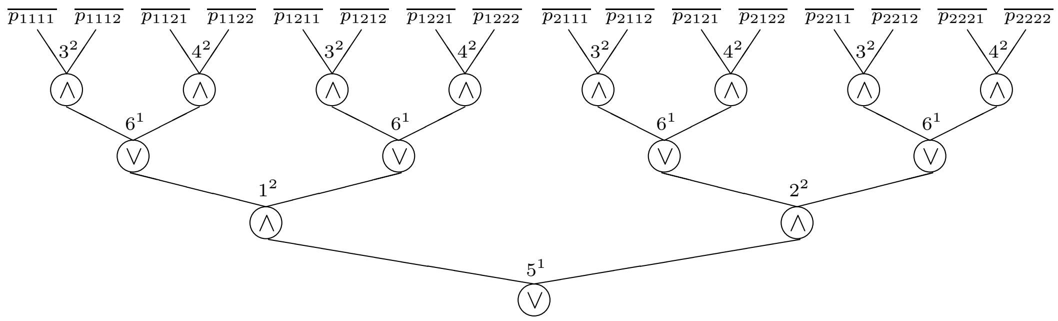
p_{1111} p_{1112} p_{1121} p_{1122} p_{1211} p_{1212} p_{1221} p_{1222} p_{2111} p_{2112} p_{2121} p_{2122} p_{2211} p_{2212} p_{2221} p_{2222}
3^1 4^1 3^1 4^1 3^1 4^1 3^1 4^1
6^2 6^2 6^2 6^2
1^1 2^1
5^2

Figure 13: $\forall x \exists y \forall z/x,y \exists t/x,y\ p(x,y,z,t)$ (when the universe is $\{1,2\}$)



$\overline{p_{1111}}$ $\overline{p_{1112}}$ $\overline{p_{1121}}$ $\overline{p_{1122}}$ $\overline{p_{1211}}$ $\overline{p_{1212}}$ $\overline{p_{1221}}$ $\overline{p_{1222}}$ $\overline{p_{2111}}$ $\overline{p_{2112}}$ $\overline{p_{2121}}$ $\overline{p_{2122}}$ $\overline{p_{2211}}$ $\overline{p_{2212}}$ $\overline{p_{2221}}$ $\overline{p_{2222}}$
3^1 4^1 3^1 4^1 3^1 4^1 3^1 4^1
6^2 6^2 6^2 6^2
1^1 2^1
5^2

Figure 14: $\neg\forall x \exists y \forall z/x,y \exists t/x,y\ p(x,y,z,t)$ (when the universe is $\{1,2\}$)



$\overline{p_{1111}}$ $\overline{p_{1112}}$ $\overline{p_{1121}}$ $\overline{p_{1122}}$ $\overline{p_{1211}}$ $\overline{p_{1212}}$ $\overline{p_{1221}}$ $\overline{p_{1222}}$ $\overline{p_{2111}}$ $\overline{p_{2112}}$ $\overline{p_{2121}}$ $\overline{p_{2122}}$ $\overline{p_{2211}}$ $\overline{p_{2212}}$ $\overline{p_{2221}}$ $\overline{p_{2222}}$
3^2 4^2 3^2 4^2 3^2 4^2 3^2 4^2
6^1 6^1 6^1 6^1
1^2 2^2
5^1

Figure 15: $\sim\forall x \exists y \forall z/x,y \exists t/x,y\ p(x,y,z,t)$ (when the universe is $\{1,2\}$)

The above cirquents are pairwise extensionally non-identical. An interpretation separating the cirquent of Figure 13 from those of Figures 14 and 15 is one that sends all atoms to $\top$. And an interpretation separating the cirquent of Figure 14 from the other two cirquents (by making the former $\top$ while the latter $\bot$) is the one that sends the four atoms p_{1111}, p_{1221}, p_{2112}, p_{2222} to $\top$ and sends all other atoms to $\bot$.

The cirquent of Figure 13 can be seen to be extensionally identical to the cirquent of figure 12. In general, the same would be the case for any pair of cirquents that syntactically relate to each other in the same way as the cirquents of Figures 13 and 12 do, namely, where one cirquent is a cirquent with all $\vee$-clusters in the lowest rank and all $\wedge$-clusters in the highest rank, and the other cirquent is the result of ignoring in the first one all $\wedge$-clusters and ignoring ranking, after which it can be understood as a cirquent in the limited sense of Section 6.

The cirquent of Figure 14 is the exact opposite of the cirquent of Figure 13, in the sense that, under any interpretation *, one is $\top$ iff the other is $\bot$. In general, if two cirquents C_1 and C_2 syntactically relate to each other as the cirquents of Figures 13 and 14 do, then, for any interpretation *, we will have $C_2^* = \neg C_1^*$, where $\neg$ is computability logic's ordinary negation operation of Definition 2.2. As for the cirquent of Figure 15, it appears to be a less natural modification of the cirquent of Figure 13 than the cirquent of Figure 14 is. In particular, it is not clear why we, in the process of transforming the cirquent of figure 13 into the cirquent of Figure 15, not only changed the label of each node to its dual, but also swapped the ranks. Furthermore, it would not be clear how to "swap" ranks if we had more than two of them. So, in spite of IF logic's tradition to see $\sim$ as the primary sort of negation and treat the "ill-behaved" $\neg$ as a second-class citizen, we come to the vision that it is $\neg$ rather than $\sim$ that is truly natural and deserves the first-class status.

Descriptions 7.4 and 7.5 only generate (sequential-gate-free) cirquents with two (in normal cases) or fewer (in pathological cases) ranks, and hence these sorts of cirquents are sufficient for capturing extended IF logic. Was there then a reasonable call for also considering cirquents with greater numbers of ranks? After all, any approach in any area of mathematics may find an infinite series of generalizations, and one should simply stop somewhere. This is true but, in the process of generalizing, one should stop only at a natural point where we have a more or less closed (in whatever sense) system. And stopping at cirquents with ≤ 2 ranks (what extended IF logic essentially did) would not be such a natural place. For, as noted earlier, the formalism of extended IF logic is not closed under its logical operators, and we would be forced to deal with a similar sort of an artificial restriction had we limited our considerations only to cirquents with ≤ 2 ranks.

To make the above point more clear, let us extend the syntax of IF logic by requiring that all occurences of quantifiers, conjunctions and disjunctions be superscripted with positive integers, satisfying the following two conditions:

- Whenever $1 \leq i < j$ and j is the superscript of some occurrence, so is i.
- Whenever i is the superscript of an occurrence of $\exists$ or $\vee$, the same i is not the superscript of an occurrence of $\forall$ or $\wedge$.

Such superscripts will be understood as indications of the ranks of the clusters originating from the corresponding occurrences of operators when turning formulas into cirquents in the style of Description 7.4. That is, clause 3 of Description 7.4 should now (for this new syntax of IF logic) read as follows:

> Impose ranking on the resulting cirquent, putting all clusters originating from occurrences of i-superscripted operators[14] into the ith rank, for any superscript i occurring in E.

We baptize this newly extended version of IF logic as **ranked IF logic**. Let us call the highest superscript appearing in a formula of ranked IF logic the **ranking depth** of that formula.

Of course, extended IF logic is the fragment of ranked IF logic limited to formulas of ranking depth ≤ 2. Namely, each non-negated formula E of IF logic translates into ranked IF logic as the result F of adding the superscript 1 to each occurrence of $\exists$ and $\vee$, and adding the superscript 2 to each occurrence of $\forall$ and $\wedge$ — well, unless E contains no $\vee$ and $\exists$, in which case the superscript 1 rather than 2 should be added to the occurrences of $\forall$ and $\wedge$. Next, for *any* formula F of ranked IF logic (including the cases when F is obtained from E as above), $\neg F$ can be understood as an abbreviation for the result of changing in F each occurrence of each literal p to $\neg p$ and vice versa, each occurrence of $\wedge$ to $\vee$ and vice versa, and each occurrence of $\forall$ to $\exists$ and vice versa, *without* changing any superscripts in this process.

[14] That is, clusters whose gates originate from occurrences of i-superscripted operators.

With $\neg$ treated as just explained, in contrast with the situation in extended IF logic, $\neg$ can meaningfully occur anywhere in an expression of ranked IF logic. For instance, we can write

$$\exists u^1 \neg \forall^1 x \exists^2 y \forall^1 z \exists^2 t/x\ p(x,y,z,t,u),$$

which will be simply understood as an abbreviation of

$$\exists u^1 \exists^1 x \forall^2 y \exists^1 z \forall^2 t/x\ \neg p(x,y,z,t,u).$$

To get a further feel of the advantages of ranked IF logic over extended IF logic, consider the formula $\neg\exists x \forall y/x \sim p(x,y,z)$ of the latter which, in ranked IF logic, will be written as $\forall^1 x \exists^2 y/x\ p(x,y,z)$. As we are dealing with a legal and hence semantically meaningful expression of extended IF logic, we naturally want to be able to quantify it — say, existentially — over z, and also be able to arbitrarily extend the original independences — say, by making both quantifiers independent of $\exists z$ and vice versa. Alas, extended IF logic does not permit to apply quantification to a $\neg$-negated compound formula. But ranked IF logic does. Namely, with a little analysis, the formula

$$\exists^1 z \forall^2 x/z \exists^3 y/z,x\ p(x,y,z) \tag{2}$$

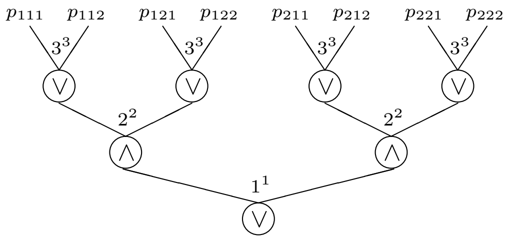
can be seen to accounts for the intuitions that we wanted to capture by $\exists z$-quantifying the formula and then making the new and old quantifiers independent of each other.

Figure 16 shows formula (2) as a cirquent, and Figure 17 shows two cirquents obtained from it by putting both existential quantifiers into the same rank in an attempt to mechanically turn (2) into an equivalent formula of ranking depth 2 (a formula of extended IF logic). Either attempt fails. Namely, let † be the interpretation that sends the two atoms p_{111}, p_{122} to $\top$ and sends all other atoms to $\bot$. Next, let ‡ be the interpretation that sends the two atoms p_{111}, p_{222} to $\top$ and sends all other atoms to $\bot$. It can be seen that

$$\big(\exists^1 z \forall^2 x/z \exists^3 y/z,x\ p(x,y,z)\big)^\dagger = \top \text{ whereas } \big(\exists^1 z \forall^2 x/z \exists^1 y/z,x\ p(x,y,z)\big)^\dagger = \bot,$$

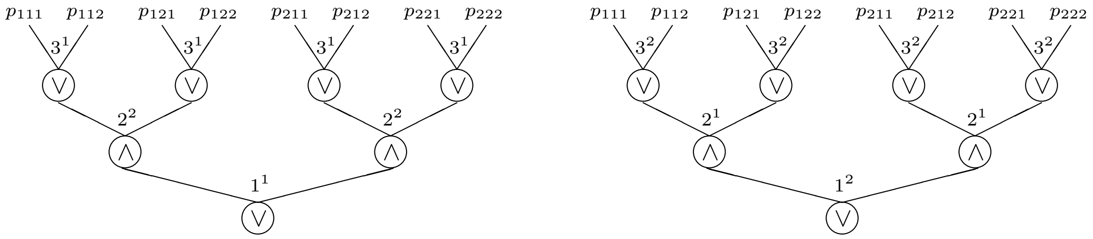
and

$$\big(\exists^1 z \forall^2 x/z \exists^3 y/z,x\ p(x,y,z)\big)^\ddagger = \bot \text{ whereas } \big(\exists^2 z \forall^1 x/z \exists^2 y/z,x\ p(x,y,z)\big)^\ddagger = \top.$$



p_{111} p_{112} p_{121} p_{122} p_{211} p_{212} p_{221} p_{222}
3^3
$\vee$
2^2
$\wedge$
1^1

Figure 16: $\exists^1 z \forall^2 x/z \exists^3 y/z,x\ p(x,y,z)$ (when the universe is $\{1,2\}$)



p_{111} p_{112} p_{121} p_{122} p_{211} p_{212} p_{221} p_{222}
3^1
2^2
1^1
3^2
2^1
1^2
$\vee$
$\wedge$

Figure 17: $\exists^1 z \forall^2 x/z \exists^1 y/z,x\ p(x,y,z)$ and $\exists^2 z \forall^1 x/z \exists^2 y/z,x\ p(x,y,z)$

As we just had a chance to observe, ranked IF logic provides greater syntactic flexibility and convenience than extended IF logic does. This gives us reasons to expect that the former is (not only a more direct and flexible means of expression but also) properly more expressive than the latter, in the same sense as the latter is known to be more expressive than ordinary, non-extended IF logic. A way to prove this conjecture can be expressing, through a formula T of ranked IF logic, a definition of truth for formulas of ranked IF logic of ranking depth ≤ 2 (such formulas fully cover extended IF logic). Now, if T itself was expressible as a formula of ranking depth ≤ 2, then we would be able to produce a paradox by writing a formula of ranking depth ≤ 2 that asserts its own not being true.

As any approach, our approach allows further generalizations. For instance, our present linear orders on ranks can be relaxed to partial orders, which may give rise to an IF-logic-style approach to independences between different ... ranks. But enough is enough. Things have already gone quite far, and further generalizations would be reasonable to make only if and when a clear call for them comes. As pointed out, a call for the generalizations (of both CoL and IF logic) we have made so far in this paper was the necessity to make the formalisms reasonably complete, and neutralize certain unsettling, incompleteness-caused phenomena such as the odd status of weak negation $\neg$ in IF logic, or the impossibility to properly develop IF logic at the purely propositional level.

However, in our step-by-step generalization process, stopping at cirquents of the present section would not be right. There is one last necessary step remaining, which will be taken in the following section.

8 General ports

Imagine a finite cirquent C in the sense of any of the previous sections. Let $p_1, \ldots, p_n$ be the atoms, listed in their lexicographic order, used (positively or negatively) in the labels of the ports of C. Then C is, in fact, an operation that takes an n-tuple of elementary games (an interpretation, that is) and produces a new, not-necessarily elementary (unless C only has $\vee$ and $\wedge$ gates) game. The same is the case when C is infinite, only here, as an operation, C may take an infinite sequence (rather than just an n-tuple) of elementary games.

Cirquents thus are systematic ways to generate and express an infinite variety of operations on games. However, as just noted, as long as we only consider cirquents in the sense of the previous section(s), all such operations are limited to ones whose arguments are elementary games $\top$ and $\bot$. These are two very special and simple cases of games. So, a natural call comes for generalizing our approach in a way that allows cirquents to express operations with not only elementary arguments, but also with arguments that can be arbitrary interactive computational problems (static games, that is) considered in CoL.

One way to achieve the above would be to change the concept of an interpretation * so that now it is allowed to send the atoms of a cirquent to any, not-necessarily-elementary, static games. By doing so we would certainly gain a lot, but just as much would be lost: the class of valid cirquents would shrink, victimizing many innocent ones such as, say, $p \rightarrow p \wedge p$ or $p \vee p \rightarrow p \sqcup p$. The point is that elementary problems (games) are meaningful and interesting in their own right, and losing the ability to differentiate them from problems in general would be too much of a sacrifice. For instance, classical logic, IF logic, or the systems of computability-logic-based arithmetic constructed in [23, 26, 27], are exclusively concerned with cases where atoms are interpreted as elementary problems.

We have a better solution. It is simply allowing two sorts of atoms in the language, one for elementary problems and the other for all problems. This way, not only do we have the ability to express combinations of problems of either sort within the same formal language, but also combinations that intermix elementary problems with not-necessarily-elementary ones.

Let us rename the objects to which we earlier refereed as (simply) "atoms" into **elementary atoms**. In addition to elementary atoms, we fix another infinite set of alphanumeric strings, disjoint from the set of elementary atoms, and call its elements **general atoms**. We shall continue using the lowercase p, q, r, s, p_1, $p(3,4)$, ... as metavariables for elementary atoms, and we will be using the uppercase P, Q, R, S, P_1, $P(3,4)$, ... as metavariables for general atoms. As before, a **literal** is L or $\neg L$, where L is an atom. In the former case the literal is said to be **positive**, and in the latter case **negative**. Such a literal will be said to be elementary or general depending on whether the atom L is elementary or general. The two literals L and $\neg L$ are said to be **opposite**. It is assumed that the question on whether a literal is elementary or general is decidable.

A **cirquent** in the sense of the present section means the same as one in the sense of the previous section, with the only difference that now not only elementary, but also general literals are allowed as labels of ports. A port is said to be *elementary* or *general*, *positive* or *negative* depending on whether its label is so. Similarly, two ports are said to be *opposite* iff their labels are so.

An **interpretation** now is a function * that (as before) sends each elementary atom p to an elementary game p^*, and sends each general atom P to any, not-necessarily-elementary, static game P^*. This function immediately extends to all literals by stipulating that, for any (elementary or general) atom W, $(\neg W)^* = \neg(W^*)$, where $\neg$ in $\neg(W^*)$ is the ordinary game negation operation of Definition 2.2.

For a run Γ and a string α, we will be using Γ^{α} to denote the result of deleting in Γ all labmoves except those that look like $\wp\alpha\beta$ (either player $\wp$ and whatever string β), and then further deleting the prefix α in each remaining move — that is, replacing each $\wp\alpha\beta$ by $\wp\beta$.

Our present definition of the legal runs of the game C^* represented by a cirquent C under an interpretation * is similar to the earlier definition(s), with the difference that now additional moves of the form $a.\alpha$ can be made, where a is (the ID of) a general port. The intuitive meaning of such a move is making the move α in the copy of the game L^* associated with a, where L is the label of a. Accordingly, the additional condition that needs to be satisfied for a legal run Γ is that, whenever a, L are as above, $\Gamma^{a.}$ (intuitively the run of L^* played in port a) should be a legal run of L^*. Below is a full definition:

Definition 8.1 Let C be a cirquent, * an interpretation, and Φ a position. Φ is a **legal position** of the game C^* iff, with "cluster" and "port" below meaning those of C, the following conditions are satisfied:

1. Every labmove of Φ has one of the following forms:
   (a) $\top c.i$, where c is a $\curlyvee$-, $\triangledown$- or $\sqcup$-cluster and i is a positive integer not exceeding the outdegree of c.
   (b) $\bot c.i$, where c is a $\curlywedge$-, $\triangle$- or $\sqcap$-cluster and i is a positive integer not exceeding the outdegree of c.
   (c) $\wp a.\beta$, where a is a general port, $\wp$ is either player, and β is some string.
2. Whenever c is a choice cluster, Φ contains at most one occurrence of a labmove of the form $\wp c.i$.
3. Whenever c is a sequential cluster and Φ is of the form $\langle \ldots, \wp c.i, \ldots, \wp c.j, \ldots \rangle$, we have $i < j$.
4. Whenever a is a general port and L is its label, $\Phi^{a.}$ is a legal position of the game L^*.

The concept of a **metaselection**, as well as the concepts **unresolved**, **resolved** and **resolvent**, for any of the eight sorts of gates, transfer from Section 7 to the present section without any changes. And metatruth (Definition 7.2) is now redefined as follows:

Definition 8.2 Let C be a cirquent, * an interpretation, Γ a legal run of C, and $\vec{f}$ a metaselection for C. In this context, with "metatrue" to be read as "**metatrue w.r.t.** $(^*, \Gamma, \vec{f})$", we say that:

- An (elementary or general) L-port a is metatrue iff $\Gamma^{a.}$ is a $\top$-won run of L^*.[15]
- A resolved gate (of any of the eight types) is metatrue iff so is its resolvent.
- No unresolved disjunctive gate (of any of the four types) is metatrue.
- Every unresolved conjunctive gate (of any of the four types) is metatrue.

Finally, we say that C is metatrue iff so is its root.

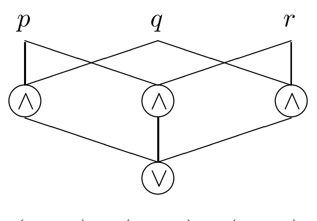
With metatruth (conservatively) redefined this way, the definition of (simply) **truth** is literally the same as before (Definition 7.3). So is the **Wn** component of the game C^* represented by a cirquent C under an interpretation *. Namely, a legal run Γ of C^* is considered to be $\top$-won iff C is true w.r.t. $(^*, \Gamma)$.

In certain cases, the elementary versus general status of atoms has no effect on validity. For instance, the cirquent $\neg p \vee p$ is valid, and so can be shown to be (whether in the weak or in the strong sense) the cirquent $\neg P \vee P$. The same does not hold for all cirquents though. An example would be $\neg p \vee (p \wedge p)$, which is valid but can be shown to be not so (whether in the strong or in the weak sense) with P instead of p. At this point we can observe the resource-consciousness of computability logic even when only parallel gates/connectives are considered: while p and $p \wedge p$ are "the same", P and $P \wedge P$ (or $P \vee P$) are not so at all: P stands for a *single* play of game P^* (whatever interpretation * we have in mind), whereas $P \wedge P$ stands for two parallel plays of P^* — that is, plays on two boards. While the game played on either board in the latter case is the same P^*, the actual runs on the two boards will not necessarily be the same (unless both players are making

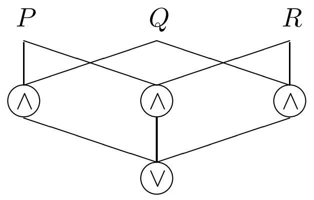
[15]Note that, if a is an elementary port, then Γ^{a} is empty, and saying that such a run is a $\top$-won run of L^* is the same as to say that $L^* = \top$.

exactly the same moves on the two boards), so that, it may well happen that the play on one board is won while on the other board is lost. Generally, winning $P \wedge P$ for $\top$ is harder than winning P, and winning P is harder than winning $P \vee P$. The situation is very different from this one with elementary atoms instead of general atoms: p is indeed "the same" as $p \wedge p$ (otherwise computability logic would not be a conservative extension of classical logic). That is because p^* is an elementary game with no moves, and hence it makes no difference whether it is "played" on one board or two boards: all of its "plays" will be identical, and hence either all of them will be won or all of them will be lost.

Earlier we pointed out the increase in expressive power of the formalism of computability logic achieved by switching from formulas to cirquents. Such a switch has an even more dramatic impact on expressive power when, along with elementary atoms, general atoms are also allowed. As we remember, finite cirquents with only $\wedge$ and $\vee$ gates and only elementary ports are not any more expressive than formulas are. The same is no the case for cirquents with general ports though. To get a feel of this, let us compare the two cirquents of Figure 18, the only difference between which is that one (on the right) has general ports where the other has elementary ports. Here and later, following our earlier practice, node IDs are omitted in these figures. Also as agreed earlier, omitted cluster IDs indicate that clustering is trivial. i.e., all clusters are singletons. Finally, omitted rank indicators should be understood as that all $\vee$-clusters are in the lowest rank and all $\wedge$-clusters are in the highest rank, even though this is, in fact, irrelevant: in the absence of nonsingleton clusters, ranking can be seen to be redundant, and how it is chosen has no effect on the semantics of the cirquent.



$(p \wedge q) \vee (p \wedge r) \vee (q \wedge r)$



No formula

Figure 18: Elementary versus general ports

The left cirquent of Figure 18 can be turned into an equivalent tree in the standard way (duplicate and separate shared nodes), yielding the formula $(p \wedge q) \vee (p \wedge r) \vee (q \wedge r)$. Everything is classical here, that is. The same trick fails for the right cirquent though. It represents a game played on three boards where $\top$, in order to win, should win on (at least) two out of the three boards. So, the cirquent that we see there is by no means equivalent to $(P \wedge Q) \vee (P \wedge R) \vee (Q \wedge R)$: the latter represents a game on six (rather than three) boards grouped into three pairs where, in order to win, $\top$ needs to win on both boards of at least one pair. This is a "two out of six" combination, which is generally easier to win than the "two out of three" combination represented by the cirquent under question. There is simply no tree-like cirquent (formula) extensionally identical, or equivalent in any reasonable weaker sense, to the right cirquent of Figure 18, meaning that expressing the game operation represented by it essentially requires the ability to account for *sharing* — the ability absent in formula-based languages. In our cirquent, each of the ports is shared between two conjunctive parents. What makes the formula $(P \wedge Q) \vee (P \wedge R) \vee (Q \wedge R)$ inadequate is that, for instance, it fails to indicate that the two occurrences of P stand for *the same copy* of game P rather than *two copies of the same game* P. And, as pointed out earlier, two copies of P are semantically not the same as just one copy.

Now we are done with defining ever more general concepts of cirquents and setting up a CoL semantics for them. The next natural step in this line of research would be elaborating deductive systems that adequately axiomatize the sets of valid cirquents. Of course, this is only possible for various subclasses of cirquents rather than all cirquents. A modest progress in this direction has already been made in [19] where a cirquent-based sound and complete system[16] was constructed. The cirquents of the language of that system have only general ports, only $\wedge$ and $\vee$ gates and, of course, no clustering and ranking.

At the formula level, very considerable advances have already been made in the direction of axiomatizing the sets of valid principles ([10]-[12], [15]-[18], [20]-[23], [24], [29]). For instance, [24] contains a sound and

[16]Soundness and completeness in [19] was proven with respect to abstract resource semantics rather than the semantics of computability logic; as we are going to see later, however, these two semantics are equivalent.

complete axiomatization for the propositional fragment of CoL with both elementary and general atoms, negation and all four — parallel, choice, sequential and toggling — sorts of conjunctions and disjunctions. Certain first-order fragments of CoL have also found sound and complete axiomatizations. The quantifiers $\sqcap, \sqcup$ turn out to be much better behaved than their classical counterparts and, in a striking difference from the latter, yield decidable first order logics. See [12], [16].

The present paper makes no axiomatization attempts, leaving this ambitious task for the future. Instead, we are going to show extensional equivalence between the semantics of CoL and its companion termed *abstract resource semantics*. This result can make the future job of axiomatizing various fragments of CoL significantly easier, as abstract resource semantics is technically much simpler and more convenient to deal with than the semantics of computability logic.

9 Abstract resource semantics

Abstract resource semantics (**ARS**) aims to formally capture our intuitions of *resources* and resource management. Resources are symmetric to *tasks*: what is a resource for the consumer, is a task for the provider. So, an equally adequate name for ARS would be "abstract task semantics".

The concept of an *atomic resource* in ARS is taken as a basic one without any definition of its nature. This makes it open to various interpretations which ARS itself, as a general-purpose tool, does not provide. The semantics of computability logic, treating atoms as variables over static games, can be seen to be one of many such possible interpretations. As for compound resources, technically their explication in ARS is given in terms of games. The formal language that ARS deals with is the same as that of computability logic. Precisely, in this paper, we let this (otherwise open-ended) language consist of all cirquents in the (most general) sense of Section 8.

There are two sorts of resources: *elementary* and *general*. Intuitively, elementary resources are "reusable" or "unexhaustable" ones, while general resources may or may not be so. Let us say in achieving a certain goal G you used the fact $2 + 2 = 4$ as an (intellectual) resource. After this usage, $2 + 2$ will still be 4, so that the resource will remain equally available for future usage if needed again. On the other hand, if you also used $\$20,000$ for achieving G, you may not be able to use the same resource again later. $2 + 2 = 4$ is an elementary resource, while $\$20,000$ is not, which makes the latter a (properly) general resource. As we may guess, elementary resources will be represented in our cirquent formalism through elementary ports, and general resources through general ports.

To get some basic intuitive feel of ARS, and to see why the latter, just like CoL, naturally calls for switching from formulas to cirquents, let us borrow a discussion from [19]. We are talking about a vending machine that has slots for 25-cent ($25c$) coins, with each slot taking a single coin. Coins can be authentic or counterfeited. Let us instead use the more generic terms *true* and *false* here, as there are various particular situations naturally and inevitably emerging in the world of resources corresponding to those two opposite values. Below are a few examples of real-world resources/tasks and the possible meanings of the two semantical values for them:

- A financial debt, which may (true) or may not (false) be eventually paid.
- An electrical outlet or a battery, which may (true) or not (false) actually have sufficient power in it.
- A standard task performed by a company's employee or an AI agent, which, eventually, may (true) or not (false) be successfully completed.
- A specified amount of computer memory required by a process, which may (true) or not (false) be available at a given time.
- A promise, which may be kept (true) or broken (false).

See Section 8 of [13] for detailed elaborations of these intuitions.

Continuing the description of our vending machine, inserting a false coin into a slot fills the slot up (so that no other coins can be inserted into it until the operation is complete), but otherwise does not fool the machine into thinking that it has received 25 cents. A candy costs 50 cents, and the machine will dispense a candy if at least two of its slots receive true coins. Pressing the "dispense" button while having inserted anything less than 50 cents, such as a single coin, or one true and two false coins, results in a non-recoverable loss.

Victor has three $25c$-coins, and he knows that two of them are true while one is perhaps false (but he has no way to tell which one is false). Could he get a candy?

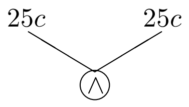
The answer depends on how many slots the machine has. Consider two cases: machine $M2$ with two slots, and machine $M3$ with three slots. Victor would have no problem with $M3$: he can insert his three coins into the three slots, and the machine, having received $\geq 50c$, will dispense a candy. With $M2$, however, Victor is in trouble. He can try inserting arbitrary two of his three coins into the two slots of the machine, but there is no guarantee that one of those two coins is not false, in which case Victor will end up with no candy and only 25 cents remaining in his pocket.

Both $M2$ and $M3$ can be understood as resources — resources turning coins into a candy. And note that these two resources are not the same: $M3$ is obviously stronger ("better"), as it allows Victor to get a candy whereas $M2$ does not, while, at the same time, anyone rich enough to be able to make $M2$ dispense a candy would be able to do the same with $M3$ as well. Yet, formulas fail to capture this important difference. $M2$ and $M3$ can be written as

$$R2 \rightarrow \mathit{Candy} \quad \text{and} \quad R3 \rightarrow \mathit{Candy},$$

respectively (with $E \rightarrow F$, as always, abbreviating $\neg E \vee F$): they consume a certain resource $R2$ or $R3$ and produce *Candy*. What makes $M3$ stronger than $M2$ is that the subresource $R3$ that it consumes is weaker (easier to supply) than the subresource $R2$ consumed by $M2$. Specifically, with one false and two true coins, Victor is able to satisfy $R3$ but not $R2$.

The resource $R2$ can be represented as the following cirquent:



which, due to being tree-like, can also be adequately written as the formula

$$25c \wedge 25c.$$

As for the resource $R3$, either one of the following two cirquents is an adequate representation of it, with one of them probably showing the relevant part of the actual physical circuitry used in $M3$:

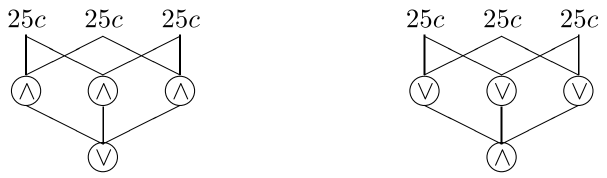


Figure 19: Two equivalent cirquents for the resource $R3$

Unlike $R2$, however, $R3$ cannot be written as a formula, for reasons similar to those that we saw when discussing Figure 18. $25c \wedge 25c$ does not fit the bill, for it represents $R2$ which, as we already agreed, is not the same as $R3$. Rewriting one of the above two cirquents — let it be the one on the right — into an "equivalent" formula in the standard way, by duplicating and separating shared nodes, results in

$$(25c \vee 25c) \wedge (25c \vee 25c) \wedge (25c \vee 25c), \tag{3}$$

which is not any more adequate than $25c \wedge 25c$. It expresses not $R3$ but the resource consumed by a machine with six coin slots grouped into three pairs, where (at least) one slot in each of the three pairs needs to receive a true coin. Such a machine thus dispenses a candy for ≥ 75 rather than ≥ 50 cents, which makes Victor's resources insufficient.

The trouble here, as in the case of the right cirquent of Figure 18, is related to the inability of formulas to explicitly account for resource sharing or the absence thereof. The right cirquent of Figure 19 stands for a conjunction of three resources, each conjunct, in turn, being a disjunction of two subresources of type $25c$. However, altogether there are three rather than six $25c$-type subresources, each one being shared between two different conjuncts of the main resource. Formula (3) is inadequate because, for example, it fails to indicate that the first and the third occurrences of "$25c$" stand for the same resource while the second and the fifth (as well as the fourth and the sixth) occurrences stand for another resource, albeit a resource of the same $25c$-type. In yet another attempt to save formulas, one could try to agree that atoms always stand for individual resources rather than resource types, then give to the three ports of the right cirquent of Figure 19 three different names P, Q, R, and represent the cirquent as the formula $(P \vee Q) \wedge (P \vee R) \wedge (Q \vee R)$. But then a crucial piece of information would be lost, specifically the information about all inputs being of the same type $25c$, as opposed to, say, the three different types $25c$, $10c$, $5c$. This would make it impossible to match Victor's resources with those inputs.

Thus, any systematic attempt to develop a logic of resources would face the necessity to go beyond formulas and use a formalism that permits to account for resource sharing, as our cirquents do. And it is an absolute shame that linear logic, commonly perceived as "the" logic of resources, does not allow to express such simple and naturally emerging combinations of resources as the "two out of three" combination expressed by the cirquents of Figure 19.

The main purpose of a good semantics should be serving as a bridge between the real world and the otherwise meaningless formal expressions of logic. And, correspondingly, the value of a semantics should be judged by how successfully it achieves this purpose, which, in turn, depends on how naturally and adequately it formalizes certain basic intuitions connecting logic with the outside world. Such intuitions behind abstract resource semantics have been amply explained and illustrated in Section 8 of [13]. The reader is strongly recommended to get familiar with that piece of literature in order to appreciate the claim of abstract resource semantics that it is a "real" semantics of resources, formalizing the resource philosophy traditionally (and, as argued in [13], somewhat wrongly) associated with linear logic and its variations. In this paper we are mainly focused on just providing formal definitions, only occasionally making brief intuitive comments, and otherwise fully relying on [13] for extended explanations of the intuitions, motivations and philosophy underlying the semantics. Even though [13] dealt with only a very modest subclass of cirquents in our present sense, the basic intuitions relevant to our treatment — at the philosophical if not technical level — were already given there.

From the technical point of view, ARS is a game semantics and, as such, only differs from the semantics of CoL in the way it treats atoms. Namely, if in CoL atoms are just placeholders which (together with the entire cirquent) become games only after an interpretation * is applied to them, ARS treats each atom as an atomic abstract resource in its own rights, and correspondingly sees the whole cirquent as a resource/task/game in its own rights, without requiring an interpretation to be applied to it. To repeat, while in CoL a cirquent C, by itself (without an interpretation) is just a formal expression but not a game, in ARS every cirquent C is directly seen as a game, which we agree to denote by $\hat{C}$. A related difference between CoL and ARS is that, in the game C^* represented by a cirquent C under a given interpretation *, CoL allows moves to be made within the games associated by * with the general literals of C. In ARS, as noted, such literals stand for atomic entities and no moves within them can or should be made. On the other hand, ARS has an additional sort of moves by $\top$, called (resource) *allocation*. Such a move looks like (a, b), where a is the ID of a P-port for some general atom P, and b is the ID of a $\neg P$-port. A condition here, called the *monogamicity condition*, is that neither a nor b could have been already used earlier in any allocation moves. As explained and illustrated in [13], the intuition behind an allocation move is that of (indeed) allocating one resource to another: a coin ($25c$) to a coin-receiving slot ($\neg 25c$), a memory ($100MB$) to a memory-requesting process ($\neg 100MB$), a power source ($100w$) to a power-consuming utensil ($\neg 100w$), an USB-interface external device (USB) to an USB port of a computer ($\neg USB$), etc. And a justification of the monogamicity condition is that if a nonelementary resource a is used by (allocated to) b, then it cannot be also used by (allocated to) another $c \neq b$.

Formally, an **allocation** for a given cirquent C is a pair (a, b) — identified with the expression "(a, b)" — where a and b are (the IDs of) general ports of C such that the label of a is P (some general atom P) and the label of b is $\neg P$.

The set of legal runs of the game associated with a cirquent in ARS is defined as follows:

Definition 9.1 Let C be a cirquent and Φ a position. Φ is a **legal position** of the game $\hat{C}$ (associated with C in ARS) iff, with "cluster" and "port" below meaning those of C, the following conditions are satisfied:

1. Every labmove of Φ has one of the following forms:

   (a) $\top c.i$, where c is a ⩛-, $\bigtriangledown$- or $\sqcup$-cluster and i is a positive integer not exceeding the outdegree of c.

   (b) $\bot c.i$, where c is a ⩚-, $\bigtriangleup$- or $\sqcap$-cluster and i is a positive integer not exceeding the outdegree of c.

   (c) $\top(a, b)$, where (a, b) is an allocation for C. This kind of a move is said to be an **allocation move**.

2. Whenever c is a choice cluster, Φ contains at most one occurrence of a labmove of the form $\wp c.i$.

3. Whenever c is a sequential cluster and Φ is of the form $\langle \ldots, \wp c.i, \ldots, \wp c.j, \ldots \rangle$, we have $i < j$.

4. Φ does not contain any two occurrences $\top(a, b)$ and $\top(a', b')$ such that $a = a'$ or $b = b'$.

By an **arrangement** for a cirquent C we mean a set $\mathcal{A}$ of allocations for C such that, whenever $(a, b), (a', b') \in \mathcal{A}$, we have either $(a, b) = (a', b')$, or both $a \neq a'$ and $b \neq b'$. We call this condition (on arrangements) the **monogamicity condition**.

Whenever C is a cirquent and Γ is a legal run of $\hat{C}$, by the **arrangement induced by** Γ we mean the set of all allocations (a, b) such that Γ contains the labmove $\top(a, b)$.

By a **situation** * for a cirquent C we mean an assignment of one of the values $\mathbb{T}$ ("*true*") or $\mathbb{F}$ ("*false*") to each port of C, satisfying the condition that, for any two elementary (but not necessarily general) ports a and b, whenever a and b have the same label, they are assigned the same value, and whenever a and b have opposite labels, they are assigned different values. Any such function * is a legitimate situation, including the cases when * assigns different values to general ports that have identical labels. Intuitively this is perfectly meaningful in the world of resources because, say, one 25c-port (slot of the vending machine) may receive a true coin while the other 25c-port may receive a false coin or no coin at all.

Note the difference between situations and interpretations. One difference is that an interpretation associates with each port a *game*, while a situation simply sends each port to one of the values $\mathbb{T}$ or $\mathbb{F}$. Of course, in the case of elementary (but by no means general) ports, this difference is not essential, as $\mathbb{T}$ can be identified with the game $\top$ and $\mathbb{F}$ with the game $\bot$. Another difference is that the domain of an interpretation is the set of *atoms* of a cirquent while the domain of a situation is the set of *ports*. This is a substantial difference, as the same atom, with or without a negation, may be the label of many different ports. But, again, this difference is not relevant if we only consider elementary ports.

Let C be a cirquent, $\mathcal{A}$ an arrangement for C, and * a situation for C. We say that * is **consistent with** $\mathcal{A}$ iff, whenever $(a, b) \in \mathcal{A}$, the situation * assigns different values to a and b.[17]

The concept of a **metaselection**, as well as the concepts **unresolved**, **resolved** and **resolvent**, for any of the eight sorts of gates, transfer from Section 8 and hence Section 7 to the present section without any changes. And metatruth (Definition 8.2) is now redefined as follows:

Definition 9.2 Let C be a cirquent, * a situation for C, Γ a legal run of $\hat{C}$, and $\vec{f}$ a metaselection for C. In this context, with "metatrue" to be read as "**metatrue w.r.t.** $(^*, \Gamma, \vec{f})$", we say that:

- An (elementary or general) L-port a is metatrue iff $L^* = \mathbb{T}$.
- A resolved gate (of any of the eight types) is metatrue iff so is its resolvent.
- No unresolved disjunctive gate (of any of the four types) is metatrue.
- Every unresolved conjunctive gate (of any of the four types) is metatrue.

Finally, we say that C is metatrue iff so is its root.

With metatruth redefined this way, the definition of (simply) **truth** is literally the same as before (Definition 7.3).

Next, where C is a cirquent and Γ is a legal run of $\hat{C}$, we say that C is **accomplished** w.r.t. Γ iff, for every situation * consistent with the arrangement induced by Γ, C is true w.r.t. $(^*, \Gamma)$.

Now, where C is a cirquent and Γ is a legal run of $\hat{C}$, Γ is considered to be a $\top$**-won** run of $\hat{C}$ iff C is accomplished w.r.t. Γ. This, together with Definition 9.1, completes our definition of the game $\hat{C}$ associated with a cirquent C in ARS.

We say that an HPM $\mathcal{M}$ **accomplishes** a cirquent C iff $\mathcal{M}$ wins the game $\hat{C}$. When such an $\mathcal{M}$ exists, the cirquent C is said to be **accomplishable**.

Accomplishability is the main semantical concept of ARS. In its philosophical spirit, it is an ARS-counterpart of the classical concept of truth. Technically, however, it is more a validity- (rather than truth-) style concept. Namely, it is similar to the concept of strong validity of computability logic.

To see an example, consider the cirquent of Figure 20.

[17]In [13], a weaker condition was adopted, according to which at least one (but possibly both) of the ports a, b should be assigned $\mathbb{T}$. It is easy to see that either condition yields the same concept of validity, so that this difference is unimportant.

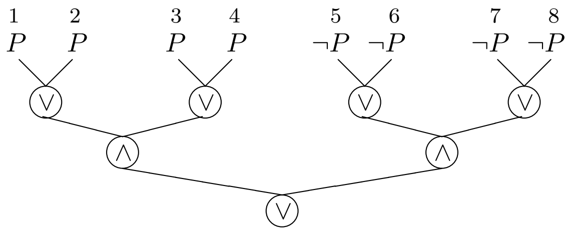


Figure 20: An accomplishable cirquent

And consider two HPMs $\mathcal{M}_1$ and $\mathcal{M}_2$ such that:

$$\mathcal{M}_1 \text{ generates the run } \Gamma_1 = \langle \top(1,5), \top(2,6), \top(3,7), \top(4,8)\rangle;$$

$$\mathcal{M}_2 \text{ generates the run } \Gamma_2 = \langle \top(1,5), \top(2,7), \top(3,6), \top(4,8)\rangle.$$

Then $\mathcal{M}_1$ does not accomplish the cirquent while $\mathcal{M}_2$ does.

Indeed, the arrangement induced by Γ_1 is

$$\mathcal{A}_1 = \{(1,5), (2,6), (3,7), (4,8)\}.$$

Let † be the situation with

$$1^\dagger = 2^\dagger = 7^\dagger = 8^\dagger = \mathbb{T}; \quad 3^\dagger = 4^\dagger = 5^\dagger = 6^\dagger = \mathbb{F}.$$

Then † is consistent with $\mathcal{A}_1$. But the cirquent is false w.r.t. $(^\dagger, \Gamma_1)$. Hence it is not accomplished w.r.t. Γ_1. Hence $\mathcal{M}_1$ does not accomplish it.

The above situation †, on the other hand, is not consistent with the arrangement

$$\mathcal{A}_2 = \{(1,5), (2,7), (3,6), (4,8)\}$$

induced by Γ_2. Moreover, with some thought, one can see that no situation that makes the cirquent of Figure 20 false can be consistent with $\mathcal{A}_2$. This means that $\mathcal{M}_2$, unlike $\mathcal{M}_1$, *does* accomplish that cirquent.

As an aside, the cirquent of Figure 20 is tree-like and hence can be written as the formula $\big((P \vee P) \wedge (P \vee P)\big) \vee \big((\neg P \vee \neg P) \wedge (\neg P \vee \neg P)\big)$. This formula, first brought forward by Blass [2] in a related context, is not provable in multiplicative linear logic or even in the extension of the latter known as *affine logic*. The same applies to the more general principle $\big((P \vee Q) \wedge (R \vee S)\big) \vee \big((\neg P \vee \neg R) \wedge (\neg Q \vee \neg S)\big)$. Thus, unlike the situation with classical logic or IF logic, the logic induced by ARS or by (the extensionally equivalent) computability logic is *not* a conservative extension of linear logic or its standard variations such as affine logic. It is this fact that makes CoL and ARS believe that linear logic is incomplete as a logic of resources.

The cirquent of Figure 21 looks very "similar" to the cirquent of Figure 20. Yet, unlike the latter, it is not accomplishable. As an exercise, the reader may want to try to understand why this is so.

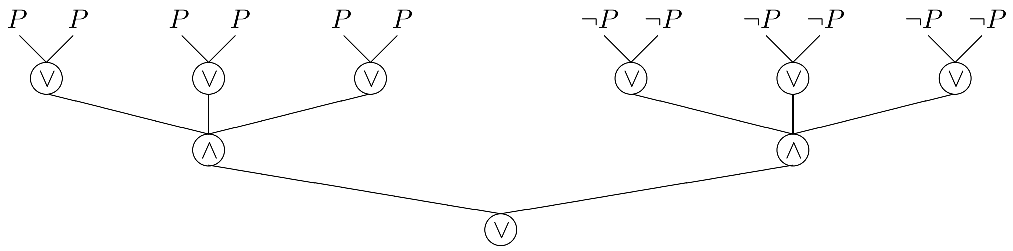


Figure 21: An unaccomplishable cirquent

To see the difference that sharing general ports may create, compare the two cirquents of Figure 22. The left cirquent can be seen to be unaccomplishable. The right cirquent only differs from the left cirquent in that ports 6 and 7 are combined together into one port 8 and shared between the two parents. This "minor" change makes it accomplishable. Namely, it is accomplished by an HPM that makes the three moves $(8,1)$, $(4,2)$ and $(5,3)$ in whatever order, i.e., sets up the arrangement $\{(8,1), (4,2), (5,3)\}$.

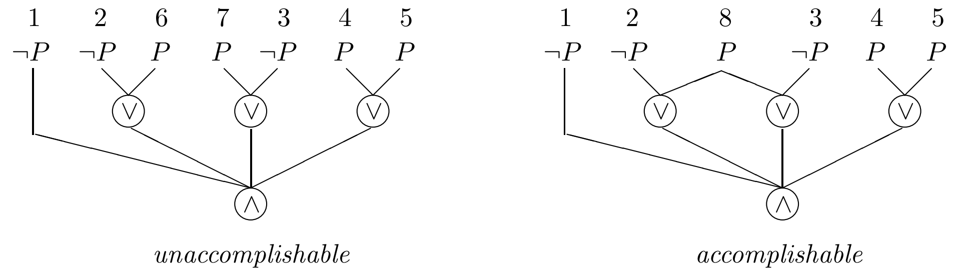

1 2 6 7 3 4 5
$\neg P$ $\neg P$ P P $\neg P$ P P
$\vee$ $\vee$ $\vee$ $\wedge$
unaccomplishable
1 2 8 3 4 5
$\neg P$ $\neg P$ P $\neg P$ P P
$\vee$ $\vee$ $\vee$ $\wedge$
accomplishable

Figure 22: Unshared versus shared general ports

Again very briefly about the intuitions behind ARS, elaborated through several papers ([8, 13, 19, 20]). Situations are full descriptions of the world in terms of what is true and what is false. What the "actual" situation is, is typically unknown to an agent (HPM) trying to accomplish a certain task (cirquent). Furthermore, the truth values of atoms may simply be indetermined before or during the process of playing the game associated with the cirquent, and can be influenced by what moves have been made. For instance, if p stands for "Victor has (or will) become a millionaire in his lifetime", the truth value of p may eventually depend on how Victor acts towards achieving p as a goal. This value is initially indetermined, for otherwise all Victor's activities would be meaningless. It will become determined only by the time when Victor dies of when the world ends.

Playing a game represented by a cirquent in ARS can be seen as resource or task management. The goal is to make sure that the eventual ("actual") situation which will determine success and over which the agent only has partial control, is favorable for the agent (guarantees a win). Resource management includes allocation decisions. The effect of each such decision/move is narrowing down the set of all possible (and otherwise unknown) situations. For instance, allocating a coin to a slot of a vending machine rules out the situation (by making it inconsistent with the arrangement that is being set up) in which the coin is true yet the slot did not receive $25c$. Resource management also includes decision-style actions associated with selectional gates or clusters. For instance, a choice between a candy and an apple that a vending machine offers to whoever inserts $50c$ into it is a $\sqcap$-combination of *Candy* and *Apple* (this is so from the machine's perspective; it becomes an $\sqcup$-combination from the user's perspective). And nature's "decisions" about whether Victor lives or dies is a $\bigtriangledown$-style combination of "*Victor is alive*" and "*Victor is dead*", where nature can switch from the former to the latter but never back, as required by the game rules associated with sequential combinations. In Section 5 we further saw resource/task intuitions associated with clustering. Remember the "Victor and Peggy in the middle of the road" example.

Historical remarks. A year before computability logic was officially born, paper [8] introduced an approach termed the *logic of tasks* which, ignoring certain unessential and flexible technical details, eventually became a conservative fragment and an ideological predecessor of both CoL and ARS. In our present terms, the language of the logic of tasks was limited to only elementary atoms (an important detail!) and formulas (tree-like cirquents) built from them using $\wedge, \vee, \sqcap, \sqcup$ in both propositional and quantifier forms. For these sorts of cirquents, the concept of validity defined in [8] coincides with our present ARS concept of accomplishability, as well as with our present CoL concept of strong validity. The above-sketched philosophical vision of the world as a set of potential situations, with the game-playing agent trying to reduce that set to favorable ones, also was originally developed within the framework of the logic of tasks. While the logic of tasks is being further studied by a number of researchers ([33], [35], [36]), the author himself abandoned it as an approach superceded by the more general computability logic.

The idea of abstract resource semantics in the proper sense (the sense that differentiates it from the more limited logic of tasks), as well as the idea of cirquents, was born in [13]. Central to this idea was considering allocations as moves in their own rights and, correspondingly, considering in the language general atoms instead of — or, rather, along with — the elementary atoms of the logic of tasks. Cirquents that [13] officially dealt with were very limited, with the root always being a $\wedge$-gate, the children of the root always being $\vee$-gates, and all grandchildren of the root being general ports. The same paper, however, outlined the possibility and expediency of considering more general sorts of cirquents and correspondingly extending the associated abstract resource semantics.

The paper [19] generalized the cirquents of [13] and the corresponding abstract resource semantics to all finite $\wedge, \vee$-cirquents with general ports. And the paper [20] outlined ARS in more or less full generality, even though for formulas only. Within that outline, Mezhirov and Vereshchagin [29] undertook a detailed study of propositional formulas in the logical signature $\{\neg, \wedge, \vee, \sqcap, \sqcup, \curlywedge, \curlyvee\}$, and proved a theorem similar to our forthcoming Theorem 10.1. The main novelty in the present extension of ARS is the idea of clustering, never considered before in either CoL or ARS. In fact, as noted earlier, cirquents, in whatever form, had never been used in CoL (as opposed to ARS) as official means of expression. And ARS only had been defined for selectional-gate-free cirquents without clustering and ranking.

Before closing this section, we want to summarize what we have already said or what the reader has probably already observed. From the technical (as opposed to, perhaps, philosophical) point of view, when cirquents with *only* elementary ports are considered, there is no difference between ARS and the semantics of CoL, as long as, in the latter, we limit our attention only to the concept of strong validity. And the above-mentioned logic of tasks is a fragments of this common part of CoL and ARS. So is classical logic, IF logic or extended IF logic. ARS and the semantics of CoL start to differ only when we consider cirquents with general ports. This difference is substantial, yet only in the *intensional* sense. The following section shows the nontrivial fact that *extensionally* there is no difference — that is, the two semantics, despite differences, still yield the same classes of valid cirquents.

10 Accomplishability and strong validity are equivalent

By a **nice game** we shall mean a game G such that, with $\wp$ (as always) standing for either player and $\neg\wp$ for the other player, we have:

- Every legal run of G is either $\langle\rangle$ or $\langle \wp m\rangle$ or $\langle \wp m, \neg\wp n\rangle$, where m, n are (the decimal representations of) some positive integers.
- The empty run $\langle\rangle$ is won by $\top$, and a legal run $\langle \wp m\rangle$ of G is won by $\wp$.
- A legal run $\langle \top m, \bot n\rangle$ of G is $\wp$-won iff so is $\langle \bot n, \top m\rangle$. This allows us to see legal runs of nice games as *sets* rather than *sequences* of labmoves, and write $\{\top m, \bot n\}$ instead of $\langle \top m, \bot n\rangle$.

Thus, different nice games differ from each other only in which runs of the form $\{\top m, \bot n\}$ are won.

By a **nice interpretation** we shall mean an interpretation that interprets each general atom as a nice game.

This section is entirely devoted to a proof of the following theorem. In it, as always, a "cirquent" means a cirquent in the most general sense defined so far, i.e., in the sense of Section 8.

Theorem 10.1 *C is strongly valid iff C is accomplishable (any cirquent C).*

Furthermore, both the "if" and "only if" parts of this theorem come in the following strong forms, respectively:

1. There is an effective procedure that takes an arbitrary HPM $\mathcal{M}$ and constructs an HPM $\mathcal{U}$ such that, if $\mathcal{M}$ accomplishes C, then $\mathcal{U}$ is a uniform solution for C.

2. If C is not accomplishable, then, for any HPM $\mathcal{U}$, there is a nice interpretation * such that $\mathcal{U}$ does not win C^*.

Before getting down to a proof of this theorem, we need to agree on certain additional details about the (otherwise loosely defined in Section 2) HPM model of computation. Namely, we assume that either tape of an HPM has a beginning but no end, with cells arranged in the left-to-right order. We also assume that, at any computation step, an HPM can make at most one move, whereas its environment can make any finite number of moves. When both players move, the order in which their moves are appended to the content of the run tape is that $\top$'s move goes before $\bot$'s moves. By a **run generated** by a given HPM $\mathcal{H}$ we mean any run that might be (depending on the environment's behavior) incrementally spelled on the run tape of $\mathcal{H}$ during its work. Thus, $\mathcal{H}$ wins a game A iff every run generated by $\mathcal{H}$ is a $\top$-won run of A.

Proof of clause 1. Consider an arbitrary cirquent C, and an arbitrary HPM $\mathcal{M}$. Our goal is to construct an HPM $\mathcal{U}$ such that, as long as $\mathcal{M}$ accomplishes C, $\mathcal{U}$ wins C^* for any interpretation *.

Understanding the idea behind our proof is not hard. We let $\mathcal{U}$ be a machine which, as far as moves associated with selectional clusters are concerned, acts the same way in C^* — i.e., makes the same selections — as $\mathcal{M}$ does in the game $\hat{C}$ associated with C in ARS. The machines $\mathcal{U}$ and $\mathcal{M}$ only differ from each other in their actions related to general ports. The strategy of $\mathcal{U}$ in general ports is that, as long as $\mathcal{M}$ has not allocated a given port to another one (or vice versa), $\mathcal{U}$ makes no moves in it. However, as soon as two general ports a and b are allocated to each other by $\mathcal{M}$, $\mathcal{U}$ starts applying copycat between the games (one being the negation of the other) associated with those ports, i.e., mimicking in either game the environment's moves made in the other game. As a result, the plays (runs) of the games associated with a and b are guaranteed to be symmetric. More precisely, one is a $\top$-delay of the other. This makes it impossible that both of those plays are lost by $\top$. We may assume that exactly one of them is won by $\top$ (if both are won, "even better"). Then, translating "lost" into the value $\mathbb{F}$ and "won" into the value $\mathbb{T}$, we get a situation ‡ for C consistent with the arrangement induced by the run Θ generated by $\mathcal{M}$. If $\mathcal{M}$ accomplishes C, the latter is true with respect to $(^\ddagger, \Theta)$. Then the game C^* can be easily seen to be won by $\mathcal{U}$.

In more technical detail, $\mathcal{U}$ works by simulating $\mathcal{M}$. For this simulation, at any step, $\mathcal{U}$ maintains a record *Configuration* for the "current" configuration of $\mathcal{M}$. The latter contains the "current" state of $\mathcal{M}$, the locations of the two scanning heads of $\mathcal{M}$, and full contents of the (imaginary) work and run tapes of $\mathcal{M}$. Initially, the state is the start state of $\mathcal{M}$, the two scanning heads are looking at the leftmost cells of their tapes, and the contents of the two tapes are empty. $\mathcal{U}$ also maintains a variable i which is initially set to 1.

After the above initialization step, $\mathcal{U}$ just acts according to the following procedure:

Procedure LOOP:

1. Using the transition function of $\mathcal{M}$ and based on the current value of *Configuration*, $\mathcal{U}$ computes the next state, next locations of the scanning heads, the next content of the work tape of $\mathcal{M}$, and correspondingly updates *Configuration*.

2. If during the above transition $\mathcal{M}$ made a move ω, $\mathcal{U}$ further updates *Configuration* by appending the labmove $\top\omega$ to the content of the imaginary run tape of $\mathcal{M}$. In addition:

   (a) If ω is not an allocation move, $\mathcal{U}$ makes the same move ω in the real play.

   (b) Suppose now ω is an allocation move (a, b). Let Υ be the longest initial segment of the position spelled on the run tape of $\mathcal{U}$ such that the last labmove of Υ, as a string spelled on the tape, ends at location j (i.e., the last symbol of the labmove is written in the jth cell) for some $j < i$. Then $\mathcal{U}$ looks up, within (but not beyond) Υ, all the labmoves $\bot a.\beta_1, \ldots, \bot a.\beta_n$ of the form $\bot a.\beta$, and makes the moves $b.\beta_1, \ldots, b.\beta_n$ in the real play. Similarly, it looks up within Υ all the labmoves $\bot b.\alpha_1, \ldots, \bot b.\alpha_m$ of the form $\bot b.\alpha$, and makes the moves $a.\alpha_1, \ldots, a.\alpha_m$.

3. Now $\mathcal{U}$ checks its run tape to see whether it contains a labmove $\bot\omega$ which, as a string spelled on the tape, ends exactly at location i. If such an ω is found, then:

   (a) If ω looks like $c.j$ where c is a selectional cluster, $\mathcal{U}$ further updates *Configuration* by adding the labmove $\bot\omega$ to the content of the imaginary run tape of $\mathcal{M}$.

   (b) If ω looks like $a.\delta$ where a is a general port already allocated by $\mathcal{M}$ to a certain port b (or vice versa) — i.e., the imaginary run tape of $\mathcal{M}$ contains the labmove $\top(a, b)$ or $\top(b, a)$ — then $\mathcal{U}$ makes the move $b.\delta$ in the real play.

4. Finally, $\mathcal{M}$ increments i to $i + 1$, and repeats LOOP.

Assume $\mathcal{M}$ accomplishes C.

Consider an arbitrary run Γ generated by $\mathcal{U}$. To this run corresponds a run Θ generated by $\mathcal{M}$ — namely, Θ is the run incrementally spelled on the imaginary run tape of $\mathcal{M}$ when the latter is simulated by $\mathcal{U}$ during playing according to the scenario of Γ. Fix these Γ and Θ. Let us also fix $\mathcal{A}$ as the arrangement induced by Θ. We further pick an arbitrary interpretation * and fix it, too.

Our assumption that $\mathcal{M}$ accomplishes C implies that $\mathcal{M}$ never makes an illegal move of $\hat{C}$ (unless its adversary does so first). We may also safely assume that $\mathcal{U}$'s environment never makes illegal moves of C^* either, for otherwise $\mathcal{U}$ automatically wins and we are done. Then a little analysis of the situation reveals that Γ is a legal run of C^* and Θ is a legal run of $\hat{C}$. We will implicitly rely on this observation in our further argument.

Consider an arbitrary pair (a, b) with $(a, b) \in \mathcal{A}$. Let P be the label of a, and thus $\neg P$ the label of b. Remember that $\Gamma^{a.}$ is the run that took place (according to the scenario of Γ) in the copy of the game P^* associated with a and, similarly, $\Gamma^{b.}$ is the run that took place in the copy of the game $\neg P^*$ associated with b. Remember also that, for a run Ω, $\neg\Omega$ means the result of negating all labels in Ω. It is clear from our description of LOOP that $\Gamma^{a.}$ is a $\top$-delay of $\neg\Gamma^{b.}$. Assume $\Gamma^{b.}$ is a $\top$-lost run of $\neg P^*$. Then, by the definition of the game negation operation, $\neg\Gamma^{b.}$ is a $\top$-won run of P^*. But then, because P^* is a static game and $\Gamma^{a.}$ is a $\top$-delay of $\neg\Gamma^{b.}$, $\Gamma^{a.}$ is a $\top$-won run of P^*. To summarize, we have:

$$\begin{array}{l}\textit{Suppose } (a,b)\in\mathcal{A}, \textit{ and } P \textit{ and } \neg P \textit{ are the labels of } a \textit{ and } b. \textit{ Then}\\ \textit{either } \Gamma^{a.} \textit{ is a } \top\textit{-won run of } P^*, \textit{ or } \Gamma^{b.} \textit{ is a } \top\textit{-won run of } \neg P^*, \textit{ or both.}\end{array} \tag{4}$$

We define a situation † for C by stipulating that, for any elementary or general L-port c of C, $c^\dagger = \mathbb{T}$ iff $\Gamma^{c.}$ is a $\top$-won run of L^*.

From our description of the work of $\mathcal{U}$ it is clear that $\mathcal{M}$ and $\mathcal{U}$ act in exactly the same ways in the selectional clusters of C. That is, Γ and Θ contain exactly the same labmoves of the form $\wp c.j$ where c is a selectional cluster. From this observation and our choice of †, with a little thought, we can see that:

$$C \textit{ is true w.r.t. } (^\dagger,\Theta) \textit{ iff } C \textit{ is true w.r.t. } (^*,\Gamma). \tag{5}$$

Let ‡ be the situation for C that agrees with † in all cases except that, whenever $(a,b)\in\mathcal{A}$ and $a^\dagger = b^\dagger = \mathbb{T}$, we have $a^\ddagger = \mathbb{T}$ and $b^\ddagger = \mathbb{F}$. In view of the monotonicity of the truth conditions associated with all types of gates, it is clear that

$$\textit{If } C \textit{ is true w.r.t. } (^\ddagger,\Theta), \textit{ then } C \textit{ is also true w.r.t. } (^\dagger,\Theta). \tag{6}$$

Consider any $(a,b)\in\mathcal{A}$. By our choice of ‡, it is impossible that $a^\ddagger = b^\ddagger = \mathbb{T}$. And, with (4) in mind, it is also impossible that $a^\ddagger = b^\ddagger = \mathbb{F}$. Thus, ‡ assigns different values to a and b. This means that

$$^\ddagger \textit{ is consistent with } \mathcal{A}. \tag{7}$$

Since $\mathcal{M}$ accomplishes C, in view of (7), C is true w.r.t. $(^\ddagger,\Theta)$. But then, by (6) and (5), C is true w.r.t. $(^*,\Gamma)$, meaning that Γ is a $\top$-won run of C^*. But remember that Γ was an arbitrary run generated by $\mathcal{U}$ and * was an arbitrary interpretation. Thus, $\mathcal{U}$ is a uniform solution for C. It remains to make the straightforward observation that, as promised in the theorem, our construction of $\mathcal{U}$ from $\mathcal{M}$ is effective.

Proof of clause 2. Our proof here is in many but respects symmetric to the proof of clause 1.

Consider an arbitrary cirquent C, and an arbitrary HPM $\mathcal{U}$ such that $\mathcal{U}$ wins C^* for any nice interpretation *. Our goal is to construct an HPM $\mathcal{M}$ such that $\mathcal{M}$ accomplishes C.

To understand the idea behind our proof, imagine a play of $\mathcal{U}$ over C^* for whatever nice interpretation *, on which, note, the behavior of $\mathcal{U}$ does not depend. Every (legal) move in this play signifies either a move made in a selectional cluster, or a move made in a general port. Since * is nice, for each general port, either player has exactly one move that can be made there. Let us call $\mathcal{U}$'s environment *smart* if it always makes different moves in different general ports. This is the case when, say, the environment always makes the move "a" in port a. Let us assume that, in the play that we are considering, the environment is smart. The best that then $\mathcal{U}$ can do is to *match* each P-labeled port with one (but not more!) $\neg P$-labeled port — match in the sense of mimicking adversary's moves so that the two games evolve in a symmetric fashion, to guarantee that at least one of them will be eventually won. Each time such a "matching" between ports a and b is detected, we let $\mathcal{M}$ make a move that allocates a to b. Other than this, $\mathcal{M}$ plays in the same way as $\mathcal{U}$ does, by making the same moves (selections) in selectional clusters as $\mathcal{U}$ makes. We can then show that, if $\mathcal{M}$ does not accomplish C with this strategy, any falsifying situation † for C translates into certain conditions on * under which $\mathcal{U}$ has lost C^*. This, however, is impossible because, by our assumption, $\mathcal{U}$ wins C^* for *any* nice interpretation *.

In more detailed terms, $\mathcal{M}$ works by simulating $\mathcal{U}$. For this simulation, at any step, $\mathcal{M}$ maintains a record *Configuration* for the "current" configuration of $\mathcal{U}$. The latter contains the "current" state of $\mathcal{U}$, the locations of the two scanning heads of $\mathcal{U}$, and full contents of the (imaginary) work and run tapes of $\mathcal{U}$. Initially, the state is the start state of $\mathcal{U}$, the two scanning heads are looking at the leftmost cells of their tapes, and the contents of the two tapes are empty. $\mathcal{M}$ also maintains a variable i which is initially set to 1.

After the above initialization step, $\mathcal{M}$ just acts according to the following procedure:

Procedure LOOP:

1. Using the transition function of $\mathcal{U}$ and based on the current value of *Configuration*, $\mathcal{M}$ computes the next state, next locations of the scanning heads, the next content of the work tape of $\mathcal{U}$, and correspondingly updates *Configuration*.

2. If during the above transition $\mathcal{U}$ made a move ω, $\mathcal{M}$ further updates *Configuration* by appending the labmove $\top\omega$ to the content of the imaginary run tape of $\mathcal{U}$. In addition, if ω looks like $c.j$ for some selectional cluster c, $\mathcal{M}$ makes the same move ω in the real play.

3. If i is (the ID of) a general port, $\mathcal{M}$ further updates *Configuration* by appending the labmove $\bot i.i$ (a "smart environment's" move) to the content of the imaginary run tape of $\mathcal{U}$.

4. Next, $\mathcal{M}$ checks if there is a pair (a,b) of opposite general ports with a positive and b negative, such that the imaginary run tape of $\mathcal{U}$ contains the four labmoves $\bot a.a$, $\bot b.b$, $\top b.a$, $\top a.b$, and $\mathcal{M}$ has not yet made the allocation move (a,b) in the real play. This is to what we earlier referred as "detecting a match between a and b". If such a pair (a,b) is found, $\mathcal{M}$ makes the move (a,b) in the real play.

5. Now $\mathcal{M}$ checks its run tape to see whether it contains a labmove $\bot\omega$ which, as a string spelled on the tape, ends at location i. If such an ω is found and it looks like $c.j$ where c is a selectional cluster, $\mathcal{M}$ further updates *Configuration* by appending $\bot\omega$ to the content of the imaginary run tape of $\mathcal{U}$.

6. Finally, $\mathcal{M}$ increments i to $i+1$, and repeats LOOP.

Consider an arbitrary run Θ generated by $\mathcal{M}$. To this run corresponds a run Γ generated by $\mathcal{U}$ — namely, Γ is the run incrementally spelled on the imaginary run tape of $\mathcal{U}$ when the latter is simulated by $\mathcal{M}$ during playing according to the scenario of Θ. Fix these Θ and Γ. Let us also fix $\mathcal{A}$ as the arrangement induced by Θ. We further pick an arbitrary situation † for C consistent with $\mathcal{A}$ and fix it, too.

Our assumption that $\mathcal{U}$ wins C^* for any nice interpretation * implies that $\mathcal{U}$ never makes an illegal move of C^* (unless its environment does so first). We may also safely assume that $\mathcal{M}$'s adversary never makes illegal moves of $\hat{C}$ either, for otherwise $\mathcal{M}$ automatically wins and we are done. Then, a little analysis of the situation reveals that Θ is a legal run of $\hat{C}$ and Γ is a legal run of C^* (for whatever nice interpretation *). We will implicitly rely on this observation in our further argument.

We shall say that a general port a of C is **unmatched** iff for no b does $\mathcal{A}$ contain (a,b) or (b,a). Otherwise a is **matched**, and the port b for which $\mathcal{A}$ contains (a,b) or (b,a) is said to be the **match** of a.

Let ‡ be the situation for C which agrees with † on all elementary and matched general ports, and sends each unmatched general port to $\mathbb{F}$. Obviously ‡, just like †, is consistent with $\mathcal{A}$. And, as † sends to $\mathbb{T}$ any port that ‡ does, in view of the monotonicity of all truth conditions associated with gates, it is clear that

$$\textit{If } C \textit{ is true w.r.t. } (^\ddagger,\Theta)\textit{, then so is it w.r.t. } (^\dagger,\Theta). \tag{8}$$

We now choose a nice interpretation * such that:

- For any elementary atom p, $p^* = \top$ iff there is a p-labeled (resp. $\neg p$-labeled) port a such that $a^\ddagger = \mathbb{T}$ (resp. $a^\ddagger = \mathbb{F}$).
- For any general atom P, P^* is the nice game such that any legal run $\{\bot a, \top b\}$ of it is $\top$-won iff we have one of the following:
  1. a is a P-port with $a^\ddagger = \mathbb{T}$, and $\Gamma^{a.} = \{\bot a, \top b\}$.
  2. b is a $\neg P$-port with $b^\ddagger = \mathbb{F}$, and $\Gamma^{b.} = \{\top a, \bot b\}$.

We claim the following:

$$\textit{For any general } L\textit{-port } c \textit{ of } C,\ c^\ddagger = \mathbb{T} \textit{ iff } \Gamma^{c.} \textit{ is a } \top\textit{-won run of } L^*. \tag{9}$$

To verify the above claim, let us first consider the case when $L = P$ (some general atom P) and $c^\ddagger = \mathbb{T}$. Since ‡ assigns $\mathbb{T}$ only to matched general ports, c is matched. Let d be the match of c. From our description of the work of $\mathcal{M}$ it is obvious that $\Gamma^{c.} = \{\bot c, \top d\}$. And, by clause 1 of our definition of P^*, such a run is a $\top$-won run of P^*, as desired.

Next, consider the case when $L = \neg P$ and $c^\ddagger = \mathbb{T}$. Again, since ‡ assigns $\mathbb{T}$ only to matched general ports, c is matched. Let d be its match. So, $d^\ddagger = \mathbb{F}$. Note that $\Gamma^{c.} = \{\top d, \bot c\}$ and hence $\neg\Gamma^{c.} = \{\bot d, \top c\}$. By our definition of P^*, $\{\bot d, \top c\}$ can be a $\top$-won run of P^* only if either $d^\ddagger = \mathbb{T}$ (clause 1) or $c^\ddagger = \mathbb{F}$ (clause 2). But neither of these two is true in our case. So, $\neg\Gamma^{c.}$ is a $\bot$-won run of P^*. But then, by the definition of game negation, $\Gamma^{c.}$ is a $\top$-won run of $\neg P^*$, as desired.

Next, consider the case when $L = P$ and $c^\ddagger = \mathbb{F}$. Since the smart adversary of (the simulated by $\mathcal{M}$) $\mathcal{U}$ makes the move $c.c$ for each general port c, $\Gamma^{c.}$ contains the labmove $\bot c$. If it does not contain any other labmoves, $\Gamma^{c.}$ is a $\bot$-won run of P^* because the latter is a nice game. Suppose now $\Gamma^{c.} = \{\bot c, \top d\}$ for some d. So, either c is unmatched, or d its match. If c is unmatched, then $\Gamma^{d.}$ cannot be $\{\bot d, \top c\}$ and hence, by our definition of P^*, $\Gamma^{c.}$ is not a $\top$-won run of P^*. Now suppose d is the match of c. Then $d^{\ddagger} = \mathbb{T}$ which, again, makes it impossible that $\Gamma^{c.}$ is a $\top$-won run of P^*. Thus, in all cases, $\Gamma^{c.}$ is a $\bot$-won run of P^*, as desired.

Finally, consider the case when $L = \neg P$ and $c^{\ddagger} = \mathbb{F}$. As in the previous case, $\Gamma^{c.}$ contains the labmove $\bot c$. Hence, $\neg\Gamma^{c.}$ contains $\top c$. If $\neg\Gamma^{c.}$ does not contain any other labmoves, $\neg\Gamma^{c.}$ is a $\top$-won run of P^* because P^* is a nice game; then, $\Gamma^{c.}$ is a $\bot$-won run of $\neg P^*$, and we are done. Suppose now $\neg\Gamma^{c.} = \{\bot d, \top c\}$ for some d, and hence $\Gamma^{c.} = \{\top d, \bot c\}$. Then, by clause 2 of our definition of P^*, $\neg\Gamma^{c.}$ is a $\top$-won run of P^*, meaning that $\Gamma^{c.}$ is a $\bot$-won run of $\neg P^*$, as desired. Claim (9) is now proven.

By our choice of *, claim (9) is automatically true for elementary ports instead of general ports. So, we have:

$$\textit{For any (elementary or general) } L\textit{-port } c \textit{ of } C,\ c^{\ddagger} = \mathbb{T} \textit{ iff } \Gamma^{c.} \textit{ is a } \top\textit{-won run of } L^*. \tag{10}$$

From our description of the work of $\mathcal{M}$ one can see that $\mathcal{M}$ and $\mathcal{U}$ act in exactly the same ways in selectional clusters. More precisely, Θ and Γ contain exactly the same labmoves of the form $\wp c.j$ where c is a selectional cluster of C. From this observation, in conjunction with (10), it is obvious that C is true w.r.t. $(^{\ddagger}, \Theta)$ iff it is true w.r.t. $(^*, \Gamma)$. But, by our assumption, $\mathcal{U}$ wins C^* for any nice interpretation C^*. Thus, C is true w.r.t. $(^{\ddagger}, \Theta)$. Then, by (8), C is also true w.r.t. $(^{\dagger}, \Theta)$. Now, remembering that † was an arbitrary situation consistent with the arrangement induced by Θ, we find that C is accomplished w.r.t. Θ. In other words, Θ is a $\top$-won run of $\hat{C}$. Finally, remembering that Θ was an arbitrary run generated by $\mathcal{M}$, we conclude that $\mathcal{M}$ wins $\hat{C}$. In other words, $\mathcal{M}$ accomplishes C.

11 Conclusion

We have elaborated circuit-style constructs called *cirquents*, and set up two sorts of game semantics for them: the semantics of *computability logic*, and *abstract resource semantics*. The two, while substantially different, have been proven to validate the same classes of cirquents.

Cirquents, allowing us to account for sharing substructures between different parent structures, are properly more expressive and efficient means of representing various objects of study than formulas are. This fact had already been observed in the past, but only very limited classes of cirquents had been studied so far, and only abstract resource semantics (not the semantics of computability logic) had been defined for them. The present paper extended the earlier studied cirquents by adding three non-traditional pairs of conjunctive and disjunctive gates to them. An even more important innovation was generalizing gates to what we called *clusters*. This is a generalization naturally called for by advanced approaches to game logics and resource logics. Clustering further increases the expressiveness and flexibility of cirquents. Among its merits is achieving the full expressive power of *independence friendly logic* and far beyond.

This paper has been exclusively focused on semantics. Among the subsequent natural steps within the present line of research would be attempts to construct deductive systems for various fragments of the set of valid cirquents.

References

[1] S. Abramsky and R. Jagadeesan. *Games and full completeness for multiplicative linear logic.* **Journal of Symbolic Logic** 59 (1994), pp. 543–574.

[2] A. Blass. *A game semantics for linear logic.* **Annals of Pure and Applied Logic** 56 (1992), pp. 183-220.

[3] P. Hinman. **Fundamentals of Mathematical Logic**. A. K. Peters, 2005.

[4] J. Hintikka. **Logic, Language-Games and Information: Kantian Themes in the Philosophy of Logic**. Clarendon Press 1973.

[5] J. Hintikka and G. Sandu. *Game-theoretical semantics.* In: **Handbook of Logic and Language**. J. van Benthem and A ter Meulen, eds. North-Holland 1997, pp. 361-410.

[6] W. Hodges. *Compositional semantics for a language of imperfect information.* **Logic Journal of the IGPL** 5 (1997), pp. 539-563.

[7] W. Hodges. *Logics of imperfect information: why sets of assignments?* In: **Interactive Logic**. J. van Benthem, D.M. Gabbay and B. Löve, eds. Amsterdam University Press 2007, pp. 117-113.

[8] G. Japaridze. *The logic of tasks.* **Annals of Pure and Applied Logic** 117 (2002), pp. 263-295.

[9] G. Japaridze. *Introduction to computability logic.* **Annals of Pure and Applied Logic** 123 (2003), pp. 1-99.

[10] G. Japaridze. *Propositional computability logic I.* **ACM Transactions on Computational Logic** 7 (2006), No.2, pp. 302-330.

[11] G. Japaridze. *Propositional computability logic II.* **ACM Transactions on Computational Logic** 7 (2006), No.2, pp. 331-362.

[12] G. Japaridze. *From truth to computability I.* **Theoretical Computer Science** 357 (2006), pp. 100-135.

[13] G. Japaridze. *Introduction to cirquent calculus and abstract resource semantics.* **Journal of Logic and Computation** 16 (2006), pp. 489-532.

[14] G. Japaridze. *Computability logic: a formal theory of interaction.* In: **Interactive Computation: The New Paradigm**. D. Goldin, S. Smolka and P. Wegner, eds. Springer 2006, pp. 183-223.

[15] G. Japaridze. *The logic of interactive Turing reduction.* **Journal of Symbolic Logic** 72 (2007), No.1, pp. 243-276.

[16] G. Japaridze. *From truth to computability II.* **Theoretical Computer Science** 379 (2007), pp. 20-52.

[17] G. Japaridze. *Intuitionistic computability logic.* **Acta Cybernetica** 18 (2007), No.1, pp. 77–113.

[18] G. Japaridze. *The intuitionistic fragment of computability logic at the propositional level.* **Annals of Pure and Applied Logic** 147 (2007), No.3, pp.187-227.

[19] G. Japaridze. *Cirquent calculus deepened.* **Journal of Logic and Computation** 18 (2008), No.6, pp. 983-1028.

[20] G. Japaridze. *Sequential operators in computability logic.* **Information and Computation** 206 (2008), No.12, pp. 1443-1475.

[21] G. Japaridze. *In the beginning was game semantics.* In: **Games: Unifying Logic, Language and Philosophy**. O. Majer, A.-V. Pietarinen and T. Tulenheimo, eds. Springer 2009, pp. 249-350.

[22] G. Japaridze. *Many concepts and two logics of algorithmic reduction.* **Studia Logica** 91 (2009), No.1, pp. 1-24.

[23] G. Japaridze. *Towards applied theories based on computability logic.* **Journal of Symbolic Logic** 75 (2010), pp. 565-601.

[24] G.Japaridze. *Toggling operators in computability logic.* **Theoretical Computer Science** (to appear). Preprint is available at http://arxiv.org/abs/0904.3469

[25] G.Japaridze. *A logical basis for constructive systems.* Manuscript (2010) http://arxiv.org/abs/1003.0425

[26] G.Japaridze. *Introduction to clarithmetic I.* Manuscript (2010) http://arxiv.org/abs/1003.4719

[27] G.Japaridze. *Introduction to clarithmetic II.* Mnuscript (2010) http://arxiv.org/abs/1004.3236

[28] P. Lorenzen. *Ein dialogisches Konstruktivitätskriterium.* In: **Infinitistic Methods**. In: PWN, Proc. Symp. Foundations of Mathematics, Warsaw, 1961, pp. 193-200.

[29] I. Mezhirov and N. Vereshchagin. *On abstract resource semantics and computability logic.* **Journal of Computer and System Sciences** 76 (2010), pp. 356-372.

[30] G. Sandu and A. Pietarinen. *Partiality and games: propositional logic.* **Logic Journal of the IGPL** 9 (2001), No.1, pp. 107-127.

[31] M. Stevenster. *A strategic perspective on IF games.* In: **Games: Unifying Logic, Language and Philosophy**. O. Majer, A.-V. Pietarinen and T. Tulenheimo, eds. Springer 2009, pp. 101-116.

[32] T. Tulenheimo. *Independence friendly logic.* In: **Stanford Encyclopedia of Philosophy**, 2009. http://plato.stanford.edu/entries/logic-if/

[33] G. Wang and W. Xu. *From the logic of facts to the logic of tasks.* **Fuzzy Systems and Mathematics** 18 (2004), No.1, pp.1-8.

[34] J. Väänänen. *On the semantics of informational independence.* **Logic Journal of the IGPL** 10 (2002), pp. 337-350.

[35] W. Xu and Y. Jing. *Theorems in the logic of tasks.* **Fuzzy Systems and Mathematics** 20 (2006), No.6, pp. 15-20.

[36] H. Zhang and S. Li. *The description logic of tasks: from theory to practice.* **Chinese Journal of Computers** 29 (2006), No.3, pp. 488-494.